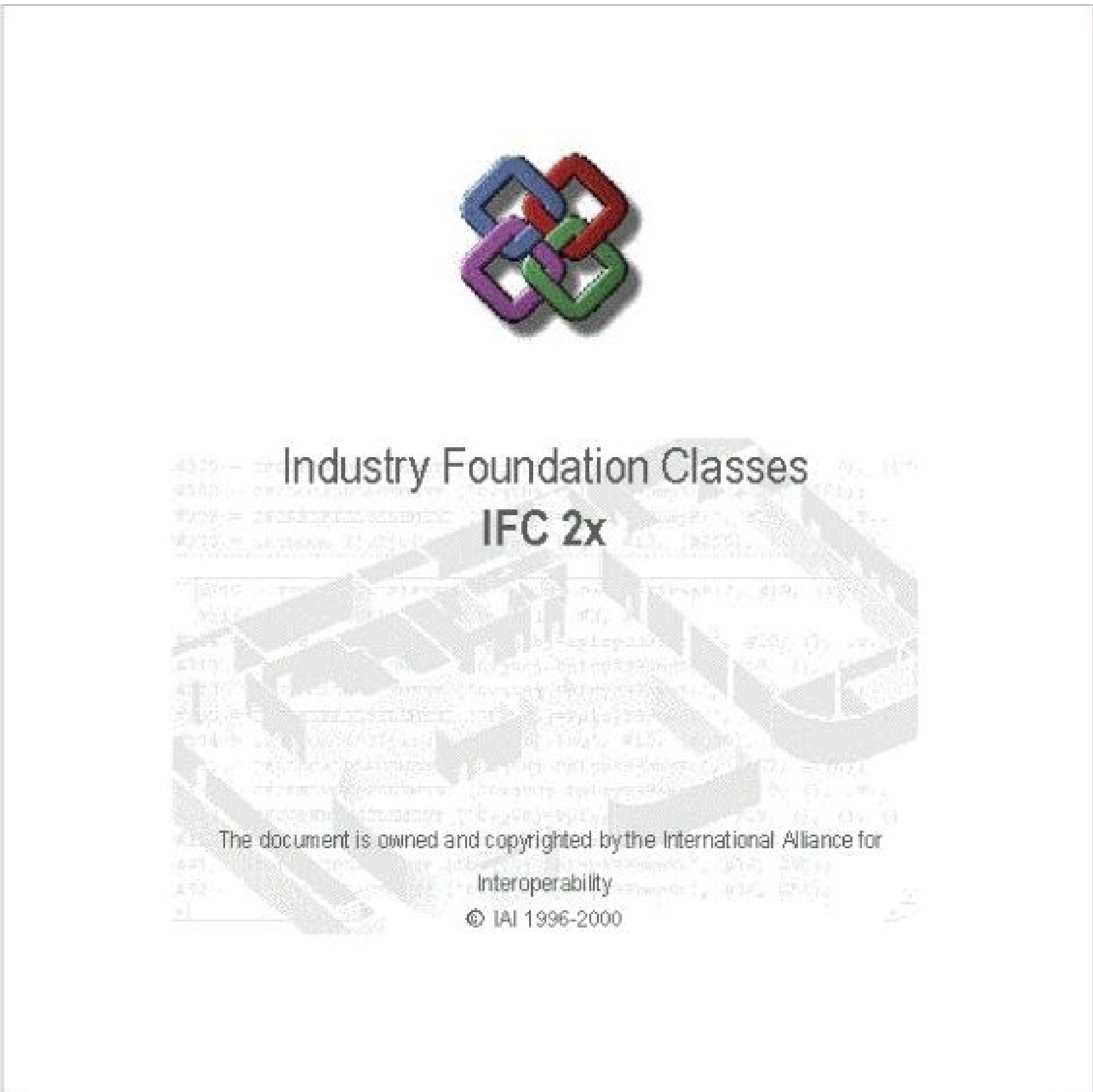

IFC 2*x*

Model Implementation Guide

International Alliance for Interoperability
Modeling Support Group
Version 1.0 of June 29, 2001

Document Control

Editor	Thomas Liebich
Development Committee	Model Support Group
Project Reference	IFC 2x Maintenance
Document Reference	IFC 2x Implementation Guide
Document Version	Version 1.0
Release Date	June 29, 2001
Status	For Comments – please send to liebich@uumail.de
Distribution	Public
Distribution format	PDF file

Acknowledgement

Part of the material for this documents had been developed by AEC3 Ltd. working at the Integrated Building Plan, Integrated Building Services (IBP/IBS) Plan Checking System carried out for the Building and Construction Authority of the Republic of Singapore within the CORENET project.

Table of Content

Table of Figures

1 Concepts defined within the Kernel

Over the IFC development timeline, starting from the early IFC Release 1.0 model and continuing with improvements during the work on Release 1.5, Release 1.5.1 and Release 2.0 and finally leading to the definitions of IFC 2x, the kernel definitions have provided the key structures of the IFC Model. These key structures are described below.

The kernel is the schema in the IFC Model that establishes the root information for all leaf node classes, i.e. those definitions that are directly used in an exchange context. Those leaf node classes utilize:

- basic object information (like identification and ownership information);
- basic relationship information to other items (like associations, aggregations);
- basic concept of type information applied to occurrence objects;
- basic concept of property information used to define the object;
- basic connection to the shape representations and the location within the project context.

To support these concepts, the key structures in the IFC kernel schema provide

- a set of generalized object class types;
- a set of generalized relationships that can occur between these class types;
- a means of extending the IFC model through sets of data (properties) declared externally.

The non-abstract descendants of the generic object class definitions (all rooting in the *IfcObject* declaration) define the leaf-node classes of the IFC model, i.e. those end-user relevant classes, that establish the meaningful data exchange. Examples are *IfcWall* (for Architecture and Coordination), *IfcSanitaryTerminal* (for HVAC design), *IfcTask* (for production planning), *IfcAsset* (for FM) and others. The relationships of these object classes to other information sets, as defined in the IFC Resource schemas, or within the sets of data declared externally, or to other object classes define the concepts (or units of functionality – UoF) applicable to the leaf-node classes.

The distinction between the leaf-node classes and the UoF provided by these classes also provides the basis for defining the IFC view definitions, those logical subsets of the IFC model, that is applicable to a certain exchange scenario or business case.

1.1 Concept Of A Root

Any leaf node class is rooted at the root entity, *IfcRoot*. It provides for the fundamental concepts of

- identification
- ownership and change information
- optional name and description attribution

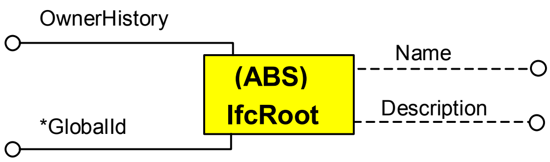

Figure 1 : Definition of IfcRoot

- GlobalId *The globally unique ID provides a unique identification to each IFC leaf note object. This is also known as a Globally Unique Identifier (GUID) or Universal Unique Identifier (UUID) by the Open Group. For the purpose of exchange in an IFC file, the GUID is compressed by an algorithm available from the IFC specification.*
- OwnerHistory *The owner history concept allows each object, relation or property to have an associated information of the creator, the creation time and about the last change applied.*
- Name, description *optional fields for attaching a name or description to the root object. However, there are subtypes for which the assigning of a name is mandatory, e.g. for a property set.*

There are three fundamental entity types in the IFC model, which are all derived from *IfcRoot*. They form the 1st level of specialization within the IFC class hierarchy.

- objects *are the generalization of any semantically treated thing (or item) within the IFC model;*
- relations *are the generalization of all relationships among things (or items) that are treaded as objectified relationships in the IFC model;*
- properties *are the generalization of all characteristics (either types or partial type, i.e. property sets) that may be assigned to objects.*

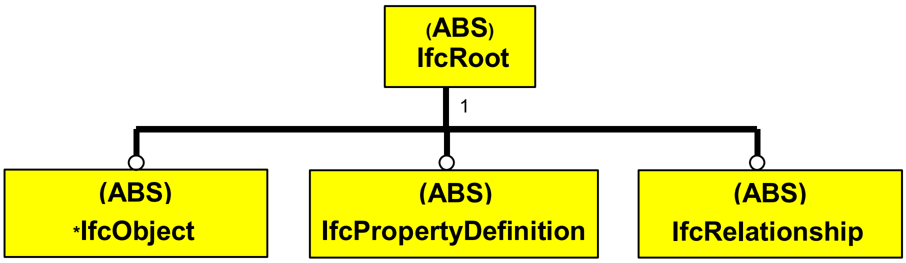

Figure 2 : Definition of the IfcRoot subtype tree

1.2 Concept Of An Object

An Object in IFC is the abstract supertype, *IfcObject*, and stands for all physically tangible items, such as wall, beam or covering, physically existing items, such as spaces, or conceptual items, such as grids or virtual boundaries. It also stands for processes, such as work tasks, for controls, such as cost items, for resources, such as labor resource, or for actors, such as persons involved in the design or construction process, etc.

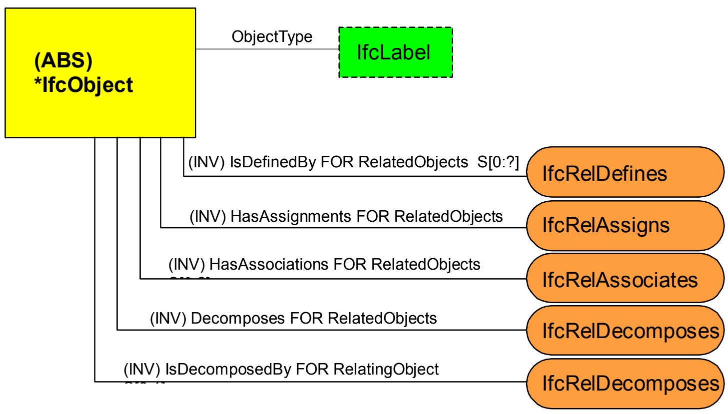

Figure 3 : Definition of IcObject

An object gets its context information from the relationships in which it is involved. The property information and, if available, the information about the underlying specific object type. An object may have an informal type descriptor assigned, which denotes a particular type to further specifies the object. The relations in which the object can be involved in have the following meaning:

- IsDefinedBy *Assignment of an object type (or object style), or sharable property set to the object, which applies the same property information to many object instances.*
- HasAssignments *Relationships to other leaf note objects. The meaning of these relationships gets further specialized at a lower hierarchy within the IFC Model.*
- HasAssociations *Relationships to concepts defined in the IFC resources, such as classification, library or document references, or material definitions.*
- Decomposes / IsDecomposedBy *Relationships that allow any object to be decomposed into its parts. Further specialization of this relationship is available.*

See section 1.5 for the next specialization level of the *IfcObject*.

1.3 Concept Of A Relationship

The concept of relationships in IFC is the relationship class *IfcRelationship*. This class represents objectification of a relationship (literally, making a relationship into a class in its own right).

The relationship class is the preferred way to handle relationships among objects. This allows relationship specific properties to be kept directly at the relationship object and enables the relationship semantics to be separated from the object attributes.

The introduction of the relationship class also allows the development of a separate subtype tree for relationship semantics.

> *NOTE: Relationship classes also occur between classes in schemata at the resource layer of the IFC Model. However, by definition of the IFC technical architecture, such classes are independent and not subtypes of IfcRelationship. Relationship classes in the resource layer are named differently to subtypes of IfcRelationship. Instead of being IfcRelXXX, they are named IfcYYYRelationship (where XXX and YYY are indicative of the semantics of the relationship).*

The link from any object class to the relationship class is handled via an inverse attribute. This convention allows to encapsulate the object class definitions, which could be distributed without the relationship objects in valid sub (or partial) models. This is a major requirement for the upcoming IFC modularization. *IfcRelationship* does not define own attributes (it subtypes does), see section 1.6 for the next specialization level of the *IfcRelationship*.

1.4 Concept Of A Property Definition

The property definition, *IfcPropertyDefinition*, is the generalization of all characteristics of objects. It reflects the specific information of an object type, versus the occurrence information of the actual object instance in the project context.

> *EXAMPLE: The style of all metal single swing windows define the common properties of panel width, thickness or material. This is defined in a subtype to IfcPropertyDefinition (IfcWindowStyle). The occurrence information, where a particular window instance is inserted (placement and assignment to the wall) is handled in the subtype of IfcObject (IfcWindow).*

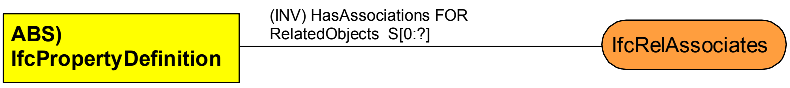

Figure 4 : Definition of IfcPropertyDefinition

The inverse attribute *HasAssociations* establishes the relation to assign classifications, library access information or document references to the property definition.

The property definition is applied to the objects using the concept of Relationships. See section 1.6.4 for the explanation of the property assignment relationship *IfcRelDefines*. See section 1.7 for the next specialization level of the *IfcPropertyDefinition*. The general concept of property assignments is explained in section 2.

1.5 Object Entity Subtype Tree

There are seven fundamental entity types in the IFC model, which are all derived from *IfcObject*. They form the 2nd level of specialization within the IFC class hierarchy under the object branch.

- products *are physical object (manufactured, supplied or created) for incorporation into a project. They may be physically existing or tangible. Products may be defined by shape representations and have a location in the coordinate space.*
- processes *are actions taking place in a project with the intent of, e.g., acquiring, constructing, or maintaining objects. Processes are placed in sequence in time.*
- controls *are concepts that control or constrain other objects. Controls can be seen as guide, specification, regulation, constraint or other requirement applied to an object that has to be fulfilled.*
- resources *are concepts that describe the use of an object mainly within a process.*

- actors *are human agents that are involved in a project during its full life cycle.*
- project *is the undertaking of some engineering activities leading towards a product.*
- group *is an arbitrary collection of objects*.

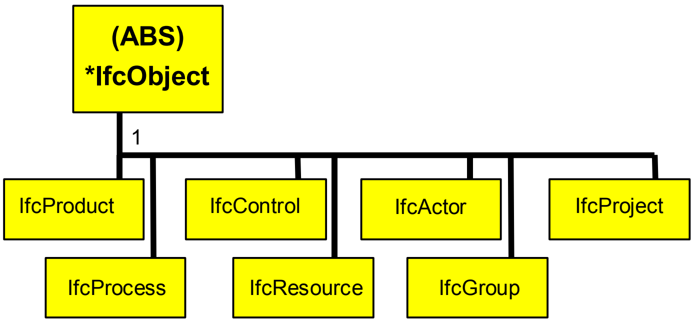

Figure 5 : Definition of the IfcObject subtype tree

1.5.1 Concept Of Product

The product type of object *IfcProduct* captures the concept of physical items that are incorporated into an AEC/FM project either directly as supplied or through construction/assembly of other products. In particular, the *IfcProduct* defines the concept of placement in some spatial context and the concept of multiple geometric representations of the product relative to that placement.

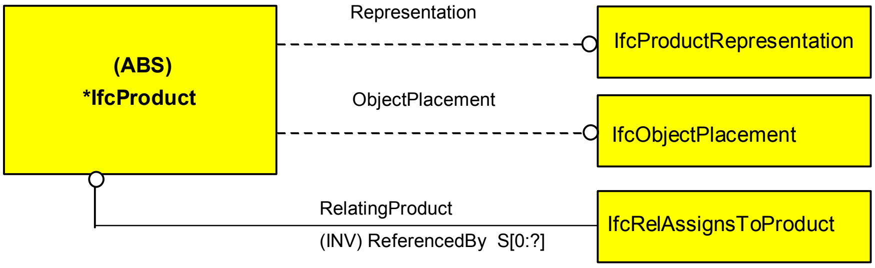

Figure 6 : Definition of IfcProduct

- ObjectPlacement *Placement of the product in the spatial context, the placement can either be absolute (relative to the world coordinate system), relative (relative to the object placement of another product), or constrained (e.g. relative to grid axes).*
- Representation *Association to the product representation, which is the container of potentially multiple geometric representations. All geometric representations of the product are defined within the ObjectPlacement, which provides the object coordinate system.*
- ReferencedBy *inverse attribute to the subtype of the general assignment relationship, which allows other objects to be assigned to products. The semantics of this relation is further declared at the level of subtypes of* IfcProduct*.*

The use of the product representation is described in section 2. A short introduction is given for the instantiable subtype of *IfcProduct*, the *IfcProxy*, below.

1.5.1.1 Concept of Proxy

The *IfcProxy* is a special subtype of product that is used for objects that are not otherwise semantically declared as part of the IFC model. A proxy may have a representation and placement (attributes inherited from *IfcProduct*) and can be further defined by property definitions.

A proxy is identified as being of one of the principal semantic types within the IFC Model and can have a tag that further qualifies its use.

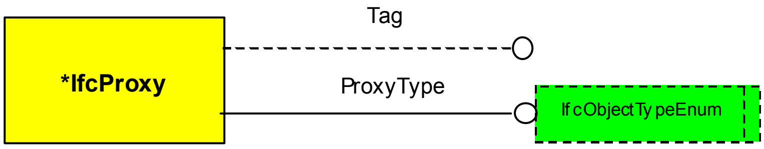

Figure 7 : Definition of IfcProxy

The *IfcProxy* is normally used to exchange objects between different computer applications, where further object information is either not available in the IFC2x specification, or within the computer applications participating in the exchange.

The *IfcProxy* is one of the few classes within the Kernel, which are not abstract types, i.e. it can be instantiated and is exchanged as a leaf note class. The *IfcProxy* could be used as a placeholder for any object type, the *ProxyType* attribute can be used to indicate the subtype of *IfcObject*, which is replaced by the *IfcProxy* object. If the *IfcProxy* stands for an *IfcProduct* (i.e. for a physical item), it may have a geometric representation. In this case, the inherited attributes *ObjectPlacement* and *Representation* needs to be given.

An *IfcProxy* with the *ProxyType* = .PRODUCT. can have any geometric representation and its *ObjectPlacement* can be relative to any element to the spatial project structure or given as absolute placement. See section 2 for more information on geometric representation.

Example:

> *The following example shows a very simple proxy, that is placed by an absolute placement within the world coordinate system of the geometric representation context. The geometric representation is given by a bounding box. There is only a single geometric shape representation, which has the type 'BoundingBox'.*

```
/* instance of the IfcProxy, concept of owner history not shown */
#10=IFCPROXY('abcdefghijklmnopqrst02', #11, 'Test proxy', $, $, #12, #13, .PRODUCT., $);

/* placement of the IfcProxy as absolute placement, since the first attribute is not
given */
#12=IFCLOCALPLACEMENT($, #14);
#14=IFCAXIS2PLACEMENT3D(#15, $, $);
#15=IFCCARTESIANPOINT((1, 1, 0));

/* single representation of the product shape, the type is 'BoundingBox' */
#13=IFCPRODUCTDEFINITIONSHAPE($, $, (#16));
#16=IFCSHAPEREPRESENTATION(#19, $, 'BoundingBox', (#17));
#17=IFCBOUNDINGBOX(#18, 3, 4, 2);
#18=IFCCARTESIANPOINT((1, 0, 0));

/* geometric representation context, establishing the world coordinate system */
#19=IFCGEOMETRICREPRESENTATIONCONTEXT($, $, 3, $, #20, $);
#20=IFCAXIS2PLACEMENT3D(#21, $, $);
#21=IFCCARTESIANPOINT((0, 0, 0));
```

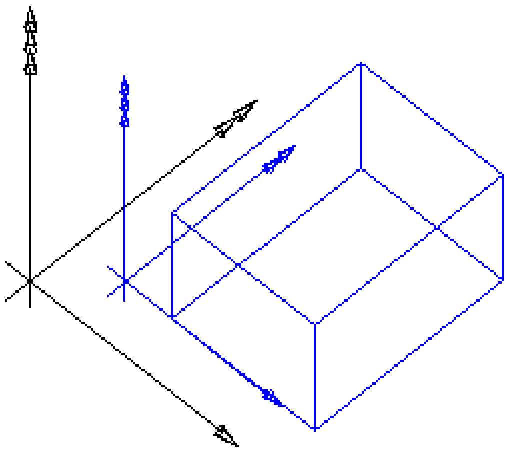

Figure 8 : Simple example of an IfcProxy

An detailed introduction into the concept of shape representation is available in section 2.

1.5.2 Concept Of Process

The process type of object *IfcProcess* captures the concept of activities undertaken within an AEC/FM project and handles the idea of work being carried out over a period of time. All processes may be named and described (by the attributes inherited from *IfcRoot*) and may incorporate a measure of productivity for the process.

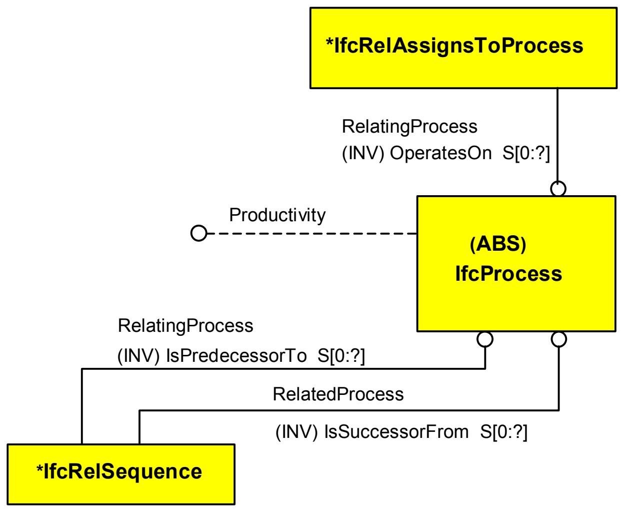

Figure 9 : Definition of IfcProcess

- OperatesOn *A process may have other objects, such as a resource or an element, as input or output, this relationship provides the ability to reference such objects*
- IsPredecessorTo *If the process is part in a sequence of processes, this relation points to the process(es) it leads to*
- IsSuccessorFrom *If the process is part in a sequence of processes, this relation points to the process(es) it originates from*

Since *IfcProcess* is an abstract entity, there cannot be direct instantiations of it. Examples of subtypes of *IfcProcess* include *IfcTask*, defined elsewhere in the IFC object model.

1.5.3 Concept Of Control

The control type of object *IfcControl* captures the concept of control or constraint applied within an AEC/FM project. *IfcControl* is an abstract supertype that has no additional attributes in IFC 2x.

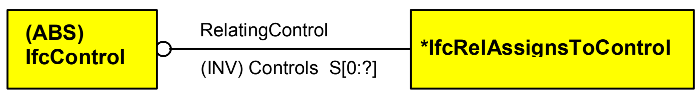

Figure 10 : Definition of IfcControl

- Controls *allows the assignment to other objects to which it is assigned as an control. By using this relation a specific type of IfcControl is assigned to a specific type of IfcObject, e.g. an* IfcCost *to an* IfcLaborResource*.*

Since *IfcControl* is an abstract entity, there cannot be direct instantiations of it. Examples of subtypes of *IfcControl* include *IfcCost, IfcWorkSchedule, IfcWorkOrder, IfcMaintenanceRecord* defined elsewhere in the IFC object model.

1.5.4 Concept Of Resource

The resource type of object *IfcResource* captures the concept of use or consumption of things within an AEC/FM project. It can be seen as the idea of things that are used within a process or by a product.

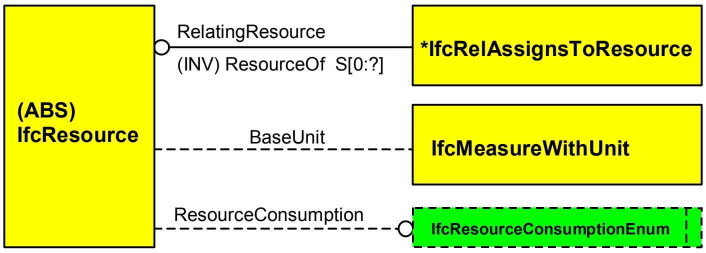

Figure 11 : Definition of IfcResource

- ResourceOf *allows to assign the resource to an product and thereby indicates it use as a resource in one or several processes.*
- BaseUnit *The basic (i.e. default, or recommended) value with associated unit that should be used for capturing the amount of measure (e.g. a volume or work periods) of the resource.*
- ResourceConsumption *An enumeration value that indicates how the resource is consumed (fully, partially or not) during its use in a process.*

The *IfcResource* refers to the definition of the product for its details and mark it as being an resource. The relationship *ResourceOf* is then used.

Since *IfcResource* is an abstract entity, there cannot be direct instantiations of it. Examples of instantiable subtypes of *IfcResource* include *IfcLaborResource, IfcConstructionEquipmentResource, IfcConstructionMaterialResource* defined elsewhere in the IFC object model.

1.5.5 Concept Of Actor

The actor type of object *IfcActor* captures the concept of people and organizations active within an AEC/FM project. It allows to indicate and share information about the participants within a project, or about the future occupants of the building within a FM context.

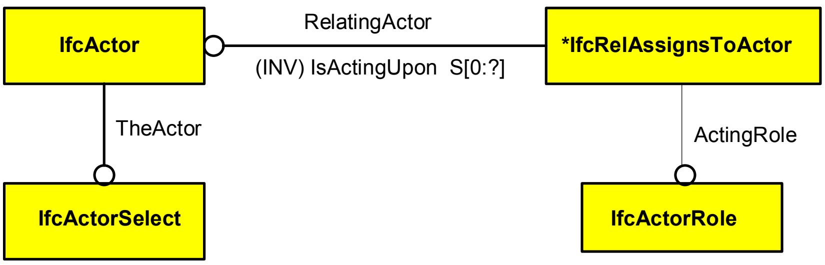

Figure 12 : Definition of IfcActor

- IsActingUpon *allows to assign any other object, e.g. a space or a resource, to the actor. It establishes the relation between the actor and some content within the project.*
- TheActor *captures the information about the actor, including name, address, phone, fax, etc.*

The *IfcActor* is non-abstract and can form part of an IFC exchange. There are more specialized types of actor available elsewhere in the IFC object model, including the *IfcOccupant*.

Example:

> *The following example shows the information about an actor, working for a general consulting firm, who is acting as the architect within an AEC project. The information "acting as the architect within the project" is captured by the relationship object IfcRelAssignsToActor, whereas the IfcActor refers to the IfcPersonAndOrganization, which holds the name and address information about the person and its organization.*

```
/* Definition of the actor, including general role and address */
#1=IFCACTOR('bacdefghijklmnopqrst01', #100, $, $, $, #3);
#3=IFCPERSONANDORGANIZATION(#4, #5, (#6));
#4=IFCPERSON($, 'Miller', 'Oscar', ('B.'), ('Dr'), $, $, $);
#5=IFCORGANIZATION($, 'General Consultants Inc.', $, $, (#7));
#6=IFCACTORROLE(.CONSULTANT., $, 'General Consultant');
#7=IFCPOSTALADDRESS(.OFFICE., $, $, $, ('500 Peachtree Street'), $, 'AnyTown', 'XX',
   '123456', 'US');

/* Definition of the project (representation context and units not shown */
#110=IFCPROJECT('bacdefghijklmnopqrst02', #101, 'Shopping Mall 123', 'Design of a new
     shopping center', $, $, $, (#111), #112);

/* Definition of the actors involvement in the project, including the specific role */
#113=IFCRELASSIGNSTOACTOR('bacdefghijklmnopqrst03', #100, $, $, (#110), .PROJECT.,
     #1, #114);
#114=IFCACTORROLE(.ARCHITECT., $, $);
```

Explanation:

The person is assigned to the organization it works for by the IfcPersonAndOrganization (see also section about the Actor concept in IFC). The actor has a role within its organization, here Consultant, this is referenced by the IfcPersonAndOrganization. If the actor is assigned to the project, it may have a more specific role, here Architect, which is assigned to the relationship IfcRelAssignsToActor.

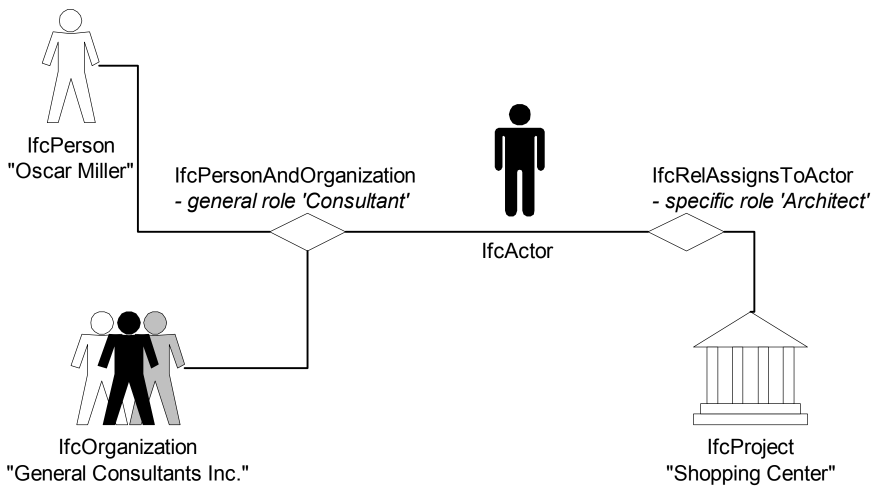

Figure 13 : Explanation to the IfcActor example

1.5.6 Concept Of Project

The project type of object *IfcProject* captures concepts that are applied generally within an AEC/FM project. A project is also the uppermost container of all information that is exchanged and shared within the AEC/FM project.

Note: *There can only be a single instance of IfcProject within an IFC file or an IFC database model. The IfcProject instance holds the global definitions for the presentation context and the units within the global context – information which can only be provided once within the IFC file.*

The *IfcProject* establishes the context of all representation within the project. It includes:

- the default global units used within the IFC file or database model

In the case of a geometric representation context it includes additionally:

- the world coordinate system
- the coordinate space dimension
- the precision used within the geometric representations, and
- optionally the indication of the true north relative to the world coordinate system

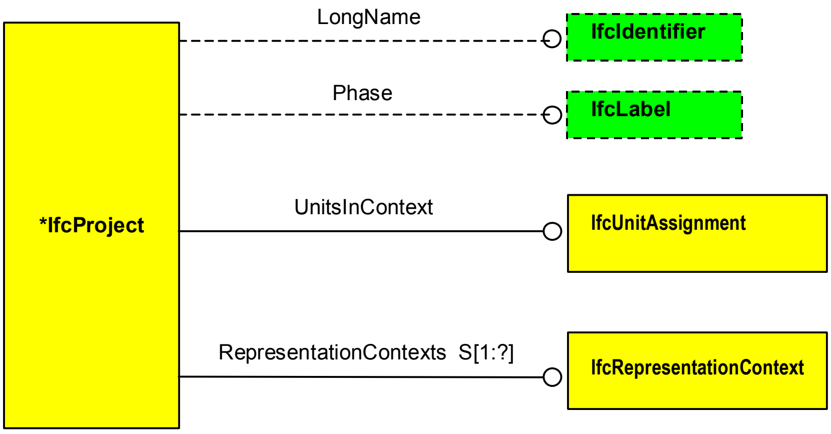

Figure 14 : Definition of IfcProject

- LongName *An unlimited string to describe the full title of the project*
- Phase *Indication of the project phase current to the exchange. The interpretation of the label has to be established by implementers agreements*
- UnitsInContext *Concept for global unit assignment, a global unit shall be defined for all defined measure types used (i.e. types within the select types IfcMeasureValue and IfcDerivedMeasureValue). Further information on the UoF "Global Unit Assignment" is given in 4.2)*
- RepresentationContexts *Concept for representation context, more than one representation context (normally a geometric representation context) can be given for a project. In case of geometric representation contexts, all have to establish the same global coordinate system.*

Example:

The project, 'Shopping Mall 123', defines the length units for its context as millimeter, and the plane angle units as radian. Thereby all length and plane angle measures within the geometry part of the project are given in millimeter and radian. Also within the non-geometry part the same units apply, if not otherwise stated. It also defines a 3D world coordinate system with the origin in 0,0,0 and a true north pointing into the direction of the positive y axis of the world coordinate system. For the precision of B-reps, a precision factor of 1.0E-06 is set.

```
/* Definition of the project (representation context and units shown below) */
#110=IFCPROJECT('bacdefghijklmnopqrst02', #101, 'Shopping Mall 123', 'Design of a new
    shopping center', $, 'Shopping Mall at 123 Main Street redevelopment project',
   'Design', (#111), #112);
#111=IFCGEOMETRICREPRESENTATIONCONTEXT($, 'Design', 3, 1.000000E-06, #115, #117);
#112=IFCUNITASSIGNMENT((#119, #118));
#115=IFCAXIS2PLACEMENT3D(#116, $, $);
#116=IFCCARTESIANPOINT((0.000000E+00, 0.000000E+00, 0.000000E+00));
#117=IFCDIRECTION((0.000000E+00, 1.000000E+00, 0.000000E+00));
#118=IFCSIUNIT(*, .LENGTHUNIT., .MILLI., .METRE.);
#119=IFCSIUNIT(*, .PLANEANGLEUNIT., $, .RADIAN.);
```

1.5.7 Concept Of Group

The group type of object *IfcGroup* captures the concept of bringing together other objects so that they can be considered as a logical unity within an AEC/FM project. A single object maybe included within several groups. A group itself maybe part of another group.

The concept of a grouping is a very general mechanism and there may be many reasons for establishing an *IfcGroup*. A Group does not define a common behavior for its members. For instance, a facilities management might require the grouping of people, furniture and various types of equipment for purposes of identification, assignment or moving.

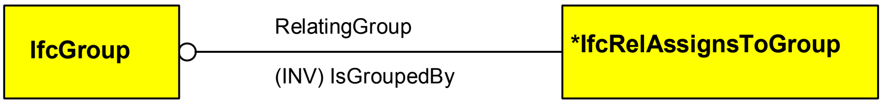

Figure 15 : Definition of IfcGroup

- IsGroupedBy *UoF for group assignment, the relationship class* IfcRelAssignsToGroup *refers to all objects, which are members of the* IfcGroup.

Example:

A group is established to make a logical collection of items (here Equipment and Furniture), which maybe handled the same way in a move process. The grouping maybe regarded as a selection process, where the select list is permanently stored and exchanged.

```
/* Definition of the group containing a furnishing and an equipment element */

#1=IFCFURNISHINGELEMENT('cbadefghijklmnopqrst10', #12, 'Executive Chair', $, 'Chair',
   $, $, 'ExCh-12345');
#7=IFCEQUIPMENTELEMENT('cbadefghijklmnopqrst02', #13, 'Free Speaker', $, 'Telephone',
   $, $, 'Tel-12345');
#8=IFCGROUP('cbadefghijklmnopqrst03', #14, 'Move Selection', $, $);
#9=IFCRELASSIGNSTOGROUP('cbadefghijklmnopqrst04', #15, $, $, (#1, #7), .PRODUCT., #8);
```

1.6 Relationship Entity Subtype Tree

There are five fundamental relationship types in the IFC model, which are all derived from *IfcRelationship*. They form the 2^{nd} level of specialization within the IFC class hierarchy under the relationship branch.

- **assignment** – is a generalization of "link" relationships among instances of objects and its various subtypes. A link denotes the specific association through which one object (the client) applies the services of other objects (the suppliers), or through which one object may navigate to other objects.
- **association** – refers to external sources of information (most notably a classification, library or document) and associates it to objects or property definitions.
- **decomposition** – defines the general concept of elements being composed or decomposed. The decomposition relationship denotes a whole/part hierarchy with the ability to navigate from the whole (the composition) to the parts and vice versa.
- **definition** – uses a type definition or property set definition (seen as partial type information) to define the properties of the object instance. It is a specific - occurrence relationship
- **connectivity** – defines the connection between two (or more) objects. The specific semantics of the connection is declared within the subtypes, defined elsewhere in the IFC model. One example is *IfcRelSequence*, that handles the concatenation of processes over time.

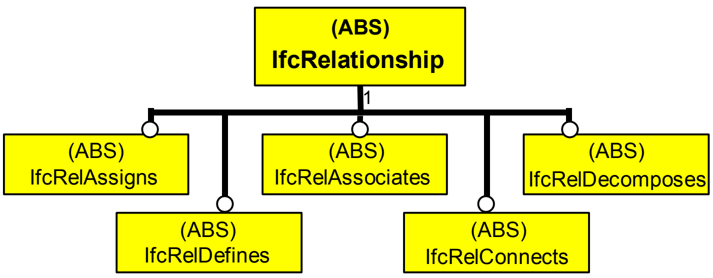

Figure 16 : Definition of IfcRelationship subtypes

1.6.1 Concept Of Assignment

The assignment relationship establishes links between objects that supply information to a specific type of object - relating to the 2nd level of specialization of the object branch. There is an assignment relationship, yielding particular semantics, for each subtype of *IfcObject*. The following assignment relationships are defined:

- assignment to products
- assignment to processes
- assignment to controls
- assignment to resources
- assignment to actors
- assignment to groups

Each of the assignment relationship subtypes relate to another of the 6 subtypes of *IfcObject* (the 7th subtype, *IfcProject*, is required to only have a single instance in a project and thus does not require a relationship to handle the assignment). Since *IfcRelAssigns* is an abstract class, only its subtype can participate in an exchange.

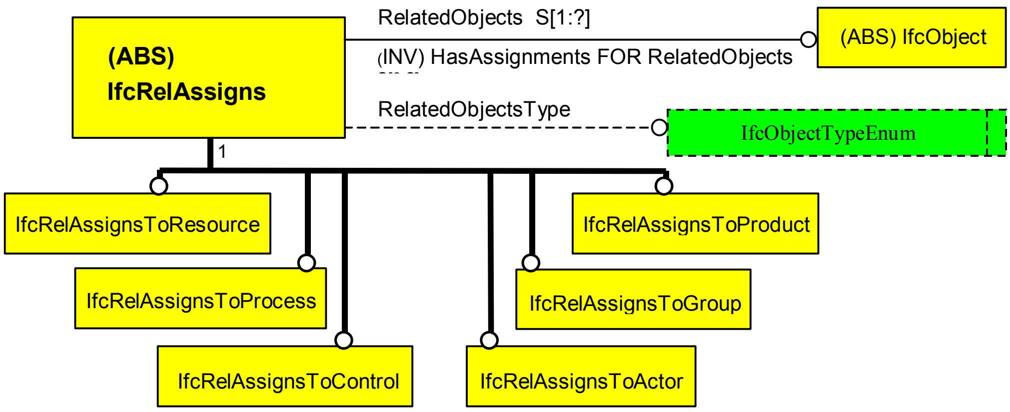

Figure 17 : Definition of IfcRelAssigns and its subtypes

The *IfcRelAssigns* relationship (and its subtypes) provide for the UoF "generic assignment" of *IfcObject*, which is redefined for the various subtypes of *IfcObject*.

1.6.1.1 Concept of Assignment to Products

Enables the assignment of one or more instances of various object types to an instance of *IfcProduct*.

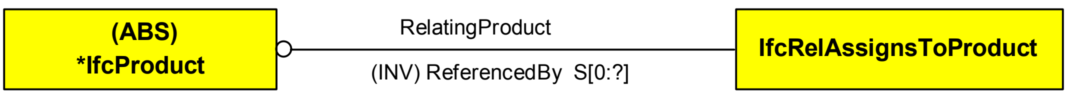

Figure 18 : Definition of IfcRelAssignsToProduct

The relationship enables the UoF "product assignment" at *IfcProduct*. An assignment of products to other objects, including other products, can be done non-hierarchically, i.e. the same product can be assigned to various objects.

Example:

> *Whereas the decomposition and spatial containment relationships (to be discussed later in the document) are required to be hierarchically, and thereby enforcing any product or element to be included in maximal one space (or spatial structure), the IfcRelAssignsToProduct could be used to assign a door to two spaces, and a space having zero to many doors assigned to.*

1.6.1.2 Concept Of Assignment To Processes

Enables the assignment of one or more instances of various object types to an instance of *IfcProcess*.

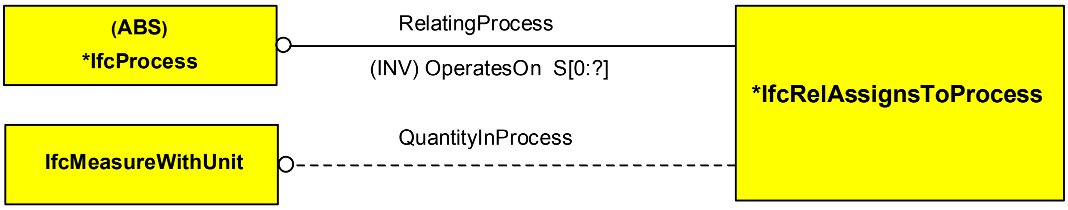

Figure 19 : Definition of IfcRelAssignsToProduct

Example:

> *The IfcRelAssignsToProcess relationship can be used to assign products or resources as either inputs or outputs to a process (e.g. as task). The relationship object would then link the various instances of IfcProduct and/or IfcResource to the IfcTask. The purpose of the assignment can be stated by the name attribute and the quantities for the assigned objects can be given by the QuantityInProcess attribute.*

1.6.1.3 Concept Of Assignment To Controls

Enables the assignment of one or more instances of various object types to an instance of *IfcControl*.

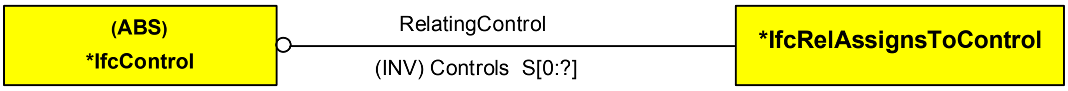

Figure 20 : Definition of IfcRelAssignsToControl

The relationship enables the UoF "control assignment" at *IfcControl*. It allows for one or several objects to have a control, like a cost, attached, or to be assigned and used by a control, like a work order, or a work plan. There are more specific subtypes of this relationship defined elsewhere in the IFC model.

Example:

> *In order to cost an object, or a collection of similar or dissimilar objects, the cost item (IfcCost being a subtype of IfcControl) is assigned to the object(s) through the relationship IfcRelCostsObjects (being a subtype of IfcRelAssignsToControl).*

1.6.1.4 Concept Of Assignment To Resources

Enables the assignment of one or more instances of various object types to an instance of *IfcResource*.

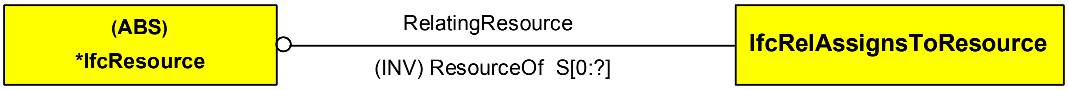

Figure 21 : Definition of IfcRelAssignsToResource

The relationship enables the UoF "resource assignment" at *IfcResource*. It allows the assignment of one or many objects being used as resources, e.g. within construction management. Examples include:

- assignment of actors to a labor or crew resource, if needed
- assignment of products to a construction product resource, where they are consumed
- assignment of equipment to a construction equipment resource, where it is used

Example:

> *Normally a labor resource would be assigned to a construction task by relating a resource (here IfcLaborResource) to an IfcTask through the IfcRelAssignsToProcess relationship. If however it is necessary to highlight a particular person or organization as the provider of the labor resource, the IfcActor (representing either a person or organization) is assigned to the IfcLaborResource by using the IfcRelAssignsToResource relationship.*

1.6.1.5 Concept Of Assignment To Actors

Enables the assignment of one or more instances of various object types to an instance of IfcActor.

The attribute *ActingRole* may be used to describe the role played by the actor in the context of the assignment. It recognizes that, for different assignments, a particular actor may take on different roles.

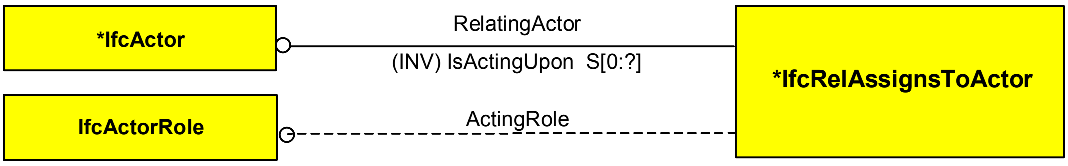

Figure 22 : Definition of IfcRelAssignsToActor

The relationship enables the UoF "actor assignment" at *IfcActor*. See section 1.5.5 for an example on actor assignment (here: defining the project team).

1.6.1.6 Concept Of Assignment To Groups

Enables the assignment of one or more instances of various object types to an instance of *IfcGroup*.

The group concept is a powerful capability within the IFC model as it allows an instance of a type of *IfcObject* to participate in various grouping concepts including (amongst other possibilities) for the purposes of costing, scheduling, asset management. An *IfcGroup* can be part of another *IfcGroup*, therefore the grouping relationship allows for nested groups.

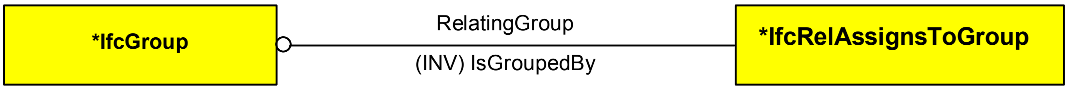

Figure 23 : Definition of IfcRelAssignsToGroup

The relationship enables the UoF "group assignment" at *IfcGroup*. See section 1.5.7 for an example on group assignment (here: definition of a group for a move process).

1.6.2 Concept Of Association

The association relationship establishes links to objects, founded within the resource part of the IFC model, that provide a reference to any type of object or property definition. Determination of whether the association is to be made to an object or to a property definition is made through the where rule at the *IfcRelAssociates*, which restricts the assignment. One of the functionality provided by the relationship is the linkage of externally defined information to objects and properties within the IFC project model.

Objects that provide a referencing service exist within the resource layer of the IFC model, such as within the *IfcExternalReferenceResource* or the *IfcMaterialResource*.

The following association relationships are defined as UoF within the *IfcKernel*:

- association with classifications
- association with documents
- association with libraries

Each is established by a specific subtype of the *IfcRelAssociates* relationship. Other subtypes include *IfcRelAssociatesMaterial*, defined elsewhere in the IFC model.

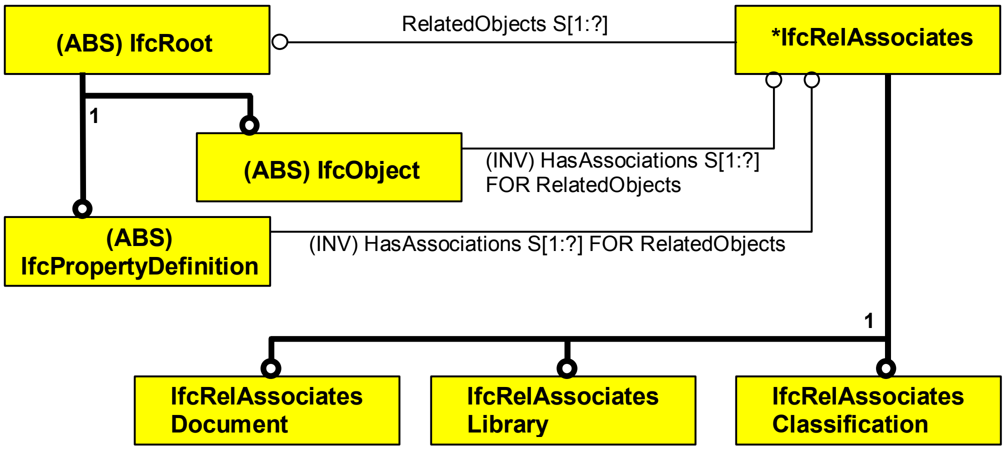

Figure 24 : Definition of IfcRelAssociates

1.6.2.1 Concept Of Association To Classification

The classification association relationship establishes links between a classification reference or notation and any type of object or property definition. It considers that any type of object contained in the *IfcObject* subtype tree, or any property definition, may be classified according to a published classification mechanism (where publication means either generally as for national and international classification systems or locally within the IFC model).

Use of the association relationship enables a single classification to be associated with many objects. By using several instances of the association relationship, a single object may also have many associated classifications if necessary. The inverse relation, *HasAssociations*, pointing to an *IfcRelAssociatesClassification* instance, provides the UoF "External Classification Association" for objects and property definitions.

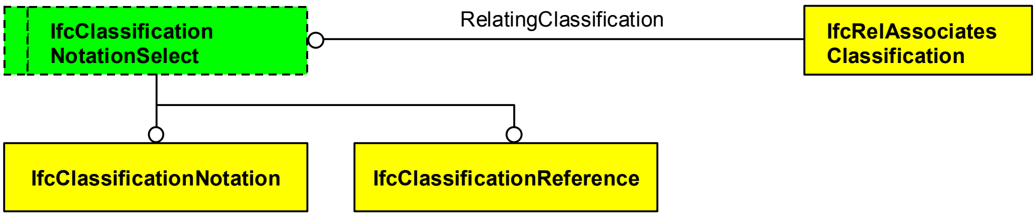

Figure 25 : Definition of IfcRelAssociatesClassification

Two different ways of classifying objects and property definitions are provided by choosing a different type from the associated *IfcClassificationNotationSelect*. By using the

> *IfcClassificationReference* a so-called "light weight" classification is provided, where only the classification notation item is given by an identification string and optional clear wording and eventually a reference to the classification system. By using the
>
> *IfcClassificationNotation* the full classification can be represented in the IFC model and the association is made to the appropriate notation of the classification facet or table.

Example:

> *The light weight classification should be used to classify a space as being of type HNF1 according to the classification DIN277. The clear wording of HNF1 is "Wohnräume" (living rooms). The classification is established by making a reference between the IfcClassificationReference "HNF1" (which only needs to appear once in the IFC model) to one or several IfcSpace objects.*

```
#1=IFCSPACE('dcbaefghijklmnopqrst01', #2, 'studio', 'open studio area', $, $, $, $,
    .ELEMENT., .INTERNAL., $);
#7=IFCCLASSIFICATIONREFERENCE($, 'HNF1', 'Wohnraeume', #8);
#8=IFCCLASSIFICATION('DIN', '', $, 'DIN277');
#9=IFCRELASSOCIATESCLASSIFICATION('dcbaefghijklmnopqrst02', #2, $, $, (#1), #7);
```

More on the concept of classification and on the full classification capabilities of IFC will be added later elsewhere to this document.

1.6.2.2 Concept Of Association To Documents

The document association relationship establishes links between a document reference or document and any type of object. It considers that any type of object contained in the *IfcObject* subtype tree or any property definition may be associated with a document. It should be noted, that the IFC model will not contain the document itself, but only the header information of the document as well as the access path to the document.

Use of the association relationship enables a single document to be associated with many objects. By using several instances of the association relationship, a single object may also have many documents associated to it, if necessary. The inverse relation, *HasAssociations*, pointing to an *IfcRelAssociatesDocument* instance, provides the UoF "External Document Association" for objects and property definitions.

Similar to the concept of classification, the external document association also provides for two levels of information. As a light weight document reference it merely associates the path to the document, normally given by the URL of the document and an optional title. As a full associated document information the level of information, as normally found within document management systems, such as author, revision, access status, confidentially, document type, summary, etc., is provided.

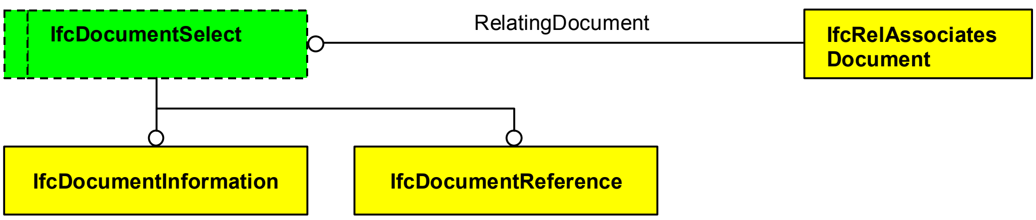

Figure 26 : Definition of IfcRelAssociatesDocument

1.6.2.3 Concept Of Association To Libraries

The library association relationship establishes links between a library reference or library and any type of object. It considers that any type of object contained in the *IfcObject* subtype tree or any property definition may be associated with a library.

Use of the association relationship enables a single library to be associated with many objects. By using several instances of the association relationship, a single object may also have many associated libraries if necessary. The inverse relation, *HasAssociations*, pointing to an *IfcRelAssociatesLibrary* instance, provides the UoF "External Library Association" for objects and property definitions.

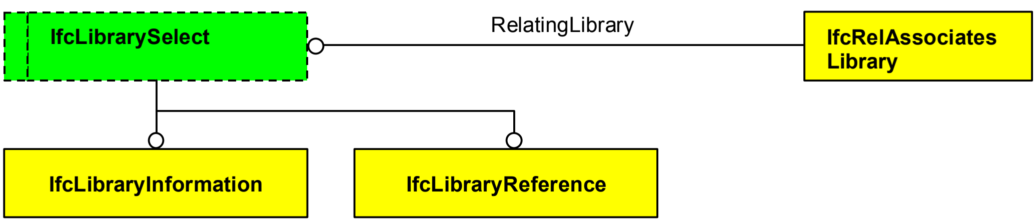

Figure 27 : Definition of IfcRelAssociatesLibrary

1.6.3 Concept Of Decomposition

The decomposition relationship establishes links between objects that participate in a whole/part relationship. It implies that there is a parent object that defines the whole (at the particular level of decomposition) and one or more child objects that define the parts. The following decomposition relationships are defined:

- aggregation
- nesting

The decomposition relationship, and any of its subtypes, is required to be hierarchical and a-cyclic. Any object can only be included in a single aggregation or nesting. However, the relationship allows for transitive decompositions, i.e. and aggregate or nesting can be part of another aggregate or nesting.

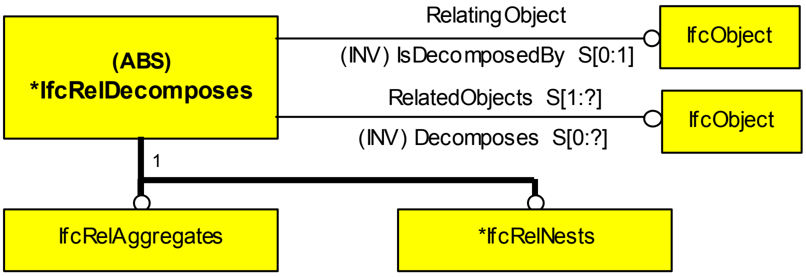

Figure 28 : Definition of IfcRelDecomposes

1.6.3.1 Concept Of Aggregation

The aggregation relationship establishes composition/decomposition relationships in which the parent (composition) object and child (decomposition) objects may all be instances of different classes within the *IfcObject* subtype tree.

1.6.3.2 Concept Of Nesting

The nesting relationship establishes composition/decomposition relationships in which the parent (composition) object and child (decomposition) objects are all instances of the same class within the *IfcObject* subtype tree.

Example:

> *A decomposition of tasks into subtasks (all instances of IfcTask) is handled by introducing the IfcRelNests relationship. The RelatingObject would point to the overall task, whereas the RelatedObjects holds the subtasks included.*

1.6.4 Concept Of Definition

The definition relationship establishes links between individual objects (also referred to as object occurrences) and type definitions (also referred to as specific objects) or property set definitions. The following definition relationships are defined:

- definition by properties
- definition by type

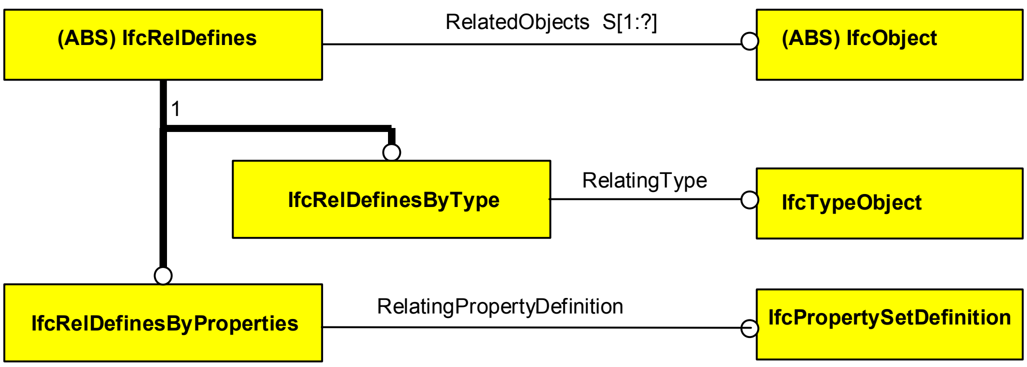

Figure 29 : Definition of IfcRelDefines

The subtypes of the abstract *IfcRelDefines* relationship allows for assigning the same type or property definition to multiple instances of an object. Thereby these objects share the same properties – this functionality is also used to share the same geometric representation among multiple object instances. In combination with the concept of mapped items, as described within the geometric representation part within section 2, it provide a block / block insertion functionality within the IFC2x model.

1.6.4.1 Concept Of Definition By Properties

The *IfcRelDefinesByProperties* relationship enables the attachment of a property set definition to any (and multiple) object instances within the *IfcObject* subtype tree.

An object may participate in several *IfcRelDefinesByProperties* relationships which enables the attachment of several property set definitions. It is more appropriate to use the *IfcRelDefinesByType* relationship for this purpose, providing that the property set definitions can logically combine into a type definition. If this is not the case, the use of several instances of the *IfcRelDefinesByProperties* relationship pointing to the same object instance may be relevant.

There are two types of property set definitions distinguished within the IFC2x model. The *IfcPropertySet* class enables the dynamic declaration of property set definitions, where the meaning of the various properties is encoded in the assigned name string. The various other subtypes of *IfcPropertySetDefinition* defined elsewhere in the IFC2x model, then statically define a fixed set of properties as a property set class. These two concepts are further explained within section 1.7.3 and 1.7.4 of this document. In addition section 2.5.1.4 introduces the property set capabilities of the IFC2x model in further detail.

1.6.4.2 Concept Of Definition By Type

The *IfcRelDefinesByType* relationship enables the attachment of an object type to any (and multiple) object instances within the *IfcObject* subtype tree.

Since an object type contains a list of property set definitions, this is the preferred approach to attaching a number of property set definitions to a single or defined set of objects. In addition, by referring to the special subtype *IfcTypeProduct*, it enables to assignment of a common shape representation, the *IfcRepresentationMap*, to multiple object instances and thus it provides for a block (or macro) insertion concept.

The two concepts, attaching multiple type related property sets and attaching additionally a representation map, are further explained within section 1.7.1 and 1.7.2 of this document. In addition section 2.5.1.4 introduces the type object capabilities of the IFC2x model in further detail.

1.6.5 Concept of Connection

The connection relationship establishes links between objects that are connected in some way. The connection may be physical or logical. Essentially the *IfcRelConnects* is a placeholder for specific relationships defined at deeper levels of the relationship class hierarchy elsewhere in the IFC2x model. The connection relationship subtypes include, e.g., *IfcRelVoidsElement, IfcRelFillsElement, IfcRelContainedInSpatialStructure, IfcRelSpaceBoundary*, or *IfcRelConnectsElements*, all defined in other schemas within the IFC2x model. The concept of

- sequence

is defined by *IfcRelSequence* within the kernel as a subtype of the connectivity relationship.

1.6.5.1 Concept Of Sequence

The sequence relationship establishes links between two processes and defines the type of linkage through the sequence enumeration data type. This enables identification of the sequence relationship in terms of the start and finish times for a process and, where a time delay occurs within the linkage, its extent.

A sequence relationship is established every time that a relationship is required between two processes (that is, sequence is a one to one relationship between processes). Thus, if a preceding process has a relationship with several succeeding processes, there is an instance of a sequence relationship between each preceding-succeeding process pair.

A rule prevents the creation of a sequence relationship between a relating and related process that are the same process.

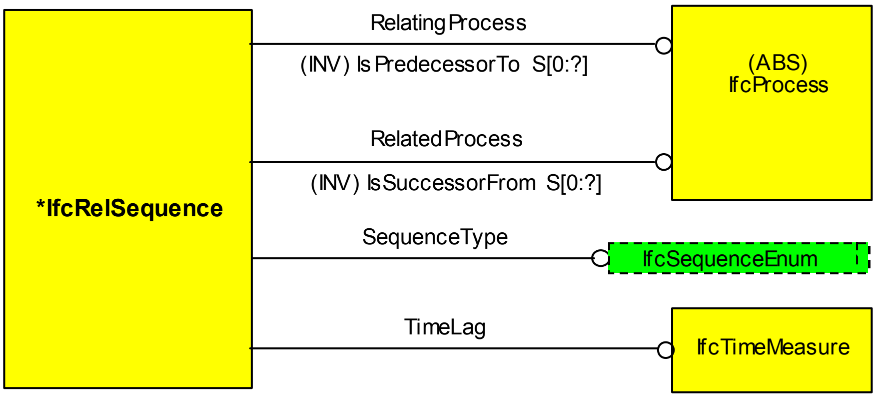

Figure 30 : Definition of IfcRelSequence

1.7 Property Definition Entity Subtype Tree

There are two fundamental concepts of property definition types in the IFC model. These are derived from *IfcPropertyDefinition*. They form the 2nd level of specialization within the IFC class hierarchy under the property definition branch.

- *type object* – defines the specific information about a type. It refers to the specific level of the well recognized generic - specific - occurrence modeling paradigm. A type defines all property information, and for the type product also all geometry information, which is common to all occurrences, that relate to that type.
- *property set definition* – defines shareable and extensible property sets attachable to occurrences of objects. The property set is regarded as a partial type information as it establishes a subset of common shared property information among occurrence objects.

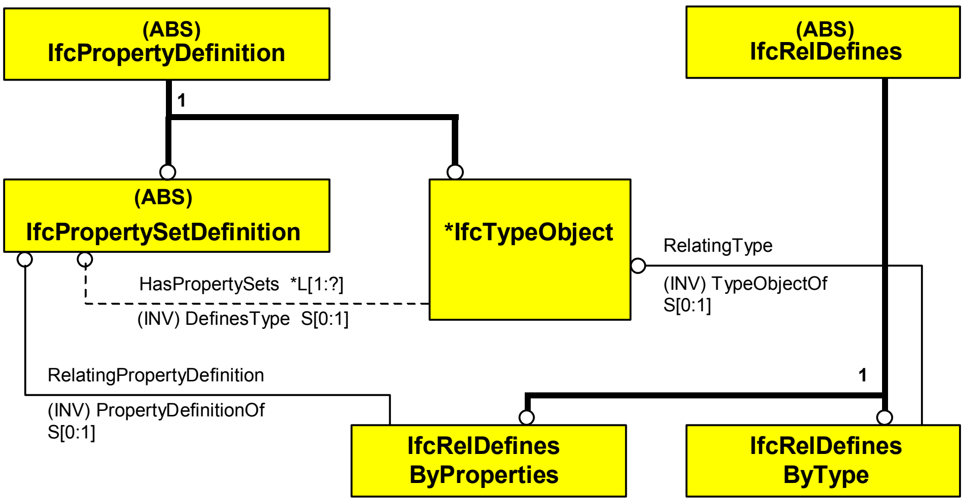

Figure 31 : Definition of IfcPropertyDefinition subtypes

1.7.1 Concept Of Type Object

The class *IfcTypeObject* expresses the concept of specification for an object (that is, it defines the specific type of an object that is otherwise generally specified by the IFC Model). It provides the means for grouping together a number of property sets that commonly work together.

1.7.2 Concept Of Type Product

The class *IfcTypeProduct*, being a subtype of *IfcTypeObject*, adds the additional capability to also define a specific type of the shape of products. It is only applicable to subtypes of *IfcProduct*, since the occurrence object instances needs to be capable to hold the object placement information, where the commonly defined representation map is located at each individual instance.

1.7.3 Concept Of Property Set Definition

The class *IfcPropertySetDefinition* expresses the concept of a set of properties that can act together to characterize an object. The set of properties providing the characterization may be declared externally to the IFC model through the *IfcPropertySet* class or within the model.

A property set may be made specific to a class through a type definition attribute of the class. The type definition mechanism is an enumeration of the possible property sets that can be attached to the class where each value in the enumeration list has a name correspondence with the property set declaration expressed through the *Name* attribute. Such property sets are published as part of the IFC model specification.

There are subtypes of *IfcPropertySetDefinition* declared elsewhere in the IFC model. They define statically defined sets of properties, expressed as attributes of those property set definition classes.

1.7.4 Concept Of Property Set

Property sets may also be defined outside of the IFC model specifications. The subtype *IfcPropertySet* allows for this dynamically defined property sets, which holds the particular semantic within the *Name* attribute. Section 2.5.1.4 describes this concept.

2 Geometric representation of products

This chapter introduces the concept of geometric representations of products within the IFC2x model. Each leaf node object in IFC that derives from *IfcProduct* can have geometric representations. The schemas holding the resource definitions are:

- *IfcGeometryResource* Defines the basic geometric representation items.
- *IfcTopologyResource* Defines the basic topological representation items.
- *IfcGeometricModelResource* Defines the geometric models to define the geometric representation of objects. It utilizes the standard geometric and topological representation items to represent geometric models.
- *IfcRepresentationResource* Defines the concepts of multiple shape representation of objects and the reference to a representation context.
- *IfcGeometryConstraintResource* Defines the concept of object placement, either using absolute or relative axis placement, or by referring to a placement relative to grid definitions.
- *IfcProfileResource* Defines standard profile (or cross section) definitions to be used in geometric representations or models of swept surfaces or swept solids.

The first three resources incorporate definitions and concepts taken from the international standard ISO/IS 10303-42 "Integrated generic resources: Geometric and topological representations". The fourth resource incorporates definitions and concepts taken from the international standard ISO/IS 10303-43 " Integrated Generic Resources – Representation Structures". The fifth and sixth resource are true additions defined within the IFC model that do not have current counterparts within the Integrated Resources as provided under ISO 10303.

2.1 Reference to the geometric representation

Any object in IFC that has a geometric representation (i.e. is a subtype of *IfcProduct*) needs to have a value for the two attributes, inherited from *IfcProduct* (see section 1.5.1). Taking *IfcWall* :

In the IFC model it is defined as:

```
ENTITY IfcWall;
   ENTITY IfcRoot;
      GlobalId      :   IfcGloballyUniqueId;
      OwnerHistory  :   IfcOwnerHistory;
      Name          :   OPTIONAL IfcLabel;
      Description   :   OPTIONAL IfcText;
   ENTITY IfcObject;
      ObjectType    :   OPTIONAL IfcLabel;
    INVERSE
         (...)
   ENTITY IfcProduct;
      ObjectPlacement  :   OPTIONAL IfcObjectPlacement;
      Representation   :   OPTIONAL IfcProductRepresentation;
    INVERSE
         (...)
   ENTITY IfcElement;
         (...)
   ENTITY IfcBuildingElement;
         (...)
END_ENTITY;  --IfcWall
```

And in the IFC file it is exchanged as:

```
#1=IFCWALL('abcdefghijklmnopqrst01', #2, $, $, $, #3, #4, $);
#3=IFCLOCALPLACEMENT($, #10);
#4=IFCPRODUCTDEFINITIONSHAPE($, $, (#11));
```

If a *Representation* is given to the subtype of *IfcProduct*, the *ObjectPlacement* has to be inserted as well. This constraint is enforced by a local rule at *IfcProduct*.

The two attributes are described below. They are defined at the level of *IfcProduct* and equally apply to all subtypes. Therefore a single concept of describing the geometric representation is applied throughout the whole IFC object model, and therefore the method of geometric representation is the same for e.g. *IfcSpace, IfcWall, IfcFlowSegment, IfcFurniture, IfcOpeningElement, IfcProxy*.

- *ObjectPlacement* Placement of the product in the spatial context, the placement can either be absolute (relative to the world coordinate system), relative (relative to the object placement of another product), or constrained (e.g. relative to grid axes).
- *Representation* Association to the product representation, which is the container of potentially multiple geometric representations. All geometric representations of the product are defined within the ObjectPlacement, which provides the object coordinate system.

The next sections describe the concept of object placement, of multiple product representation, and of the different types of shape representation in further details.

All coordinates and geometric parameters (which are length or angle measures) are unit dependent. The units common to all geometric representations within an IFC exchange are defined globally, i.e. within the context of *IfcProject* (see 1.5.6). The UoF "Global Unit Assignment" is further described within 4.2).

2.2 Concept of object placement

Any product defined as subtype of *IfcProduct* in IFC can have one or more geometric representations (e.g. a simple representation as a bounding box, and/or a complex representation as a boundary representation model). All of the geometric representations of the same object are defined within the same object coordinate system. This object coordinate system is given by the reference *ObjectPlacement* to *IfcObjectPlacement*.

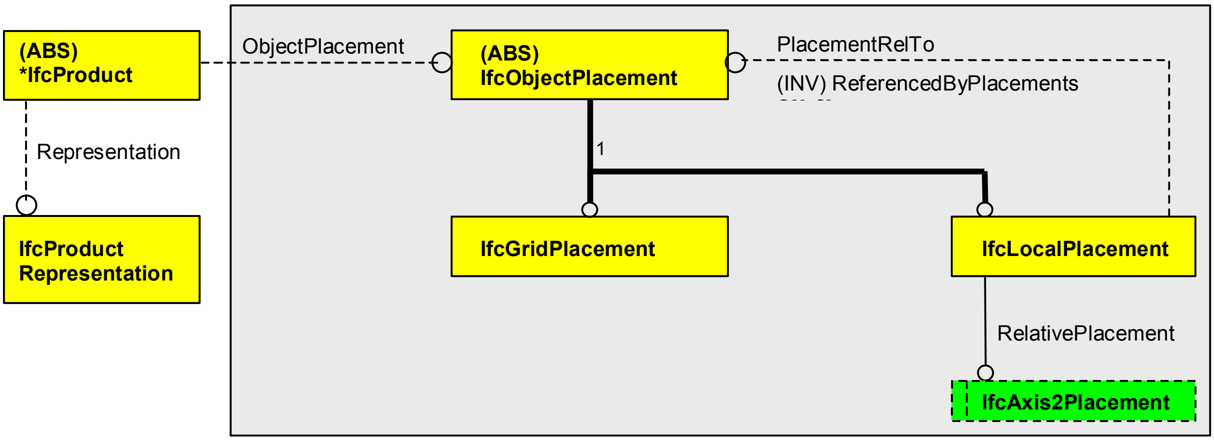

Figure 32 : Definition of IfcObjectPlacement

The *IfcObjectPlacement* has to be provided for each product, which has a shape representation defined. The entity *IfcObjectPlacement* is abstract and only the two subtypes, *IfcLocalPlacement* and *IfcGridPlacement* can be instantiated.

The object placement can be given:

- absolute, i.e. by an axis2 placement, relative to the world coordinate system,
- relative, i.e. by an axis2 placement, relative to the object placement of another product,
- by grid reference, i.e. by the virtual intersection and reference direction given by two axes of a design grid.

The object placement defines unambiguously the object coordinate system as either two-dimensional axis placement (IfcAxis2Placement2D) or three-dimensional axis placement (IfcAxis2Placement3D). The axis placement may have to be calculated, e.g. in the case of a grid placement.

2.2.1 Concept of local placement

The default way to define the placement of a product is using the relative placement, given by *IfcLocalPlacement* with an inserted value for the *PlacementRelTo* attribute. If the value of

PlacementRelTo is omitted, then the placement is given in global coordinates, i.e. within the global coordinate system, as established at the *IfcGeometricRepresentationContext*.

2.2.1.1 Concept of local relative placement

The *IfcLocalPlacement* with the *PlacementRelTo* attribute value inserted defines the relative placement of a product in relation to the placement of another product and finally through the intermediate referenced placements within the geometric representation context of the project.

Each local placement is given by an axis placement, which can be either a 2D or a 3D axis placement. The type of the axis placement shall be the same for the local relative placement and the referenced placement. The axis placement, of type *IfcAxis2Placement2D* or *IfcAxis2Placement3D,* is given by:

- the location
- the direction of the local Z-axis (in case of 3D) – if omitted always [0.,0.,1.]
- the direction within the positive XZ plane (in case of 3D) or the direction of the X-axis (in case of 2D) – if omitted always [1.,0.,0.] – or [1.,0.] for 2D

Example:

In the following example, several local placements are used, that are defined relatively. Only the placement objects, IfcLocalPlacement, are shown, but not the subtypes of IfcProduct, which utilizes the placement objects. The IfcGeometricRepresentationContext is given as it establishes the global (or world) coordinate system.

```
/* Definition of the world coordinate system */
#1=IFCGEOMETRICREPRESENTATIONCONTEXT($, '3Dmodel', 3, 1.0E-005, #2, $);
#2=IFCAXIS2PLACEMENT3D(#3, $, $);
#3=IFCCARTESIANPOINT((0.0, 0.0, 0.0));

/* Definition of the local coordinate systems */
#4=IFCLOCALPLACEMENT($, #7);
#7=IFCAXIS2PLACEMENT3D(#10, $, $);
#10=IFCCARTESIANPOINT((0.0, 0.0, 2.0));

#5=IFCLOCALPLACEMENT(#4, #8);
#8=IFCAXIS2PLACEMENT3D(#11, $, $);
#11=IFCCARTESIANPOINT((1.0, 0.0, 0.0));

#6=IFCLOCALPLACEMENT(#4, #9);
#9=IFCAXIS2PLACEMENT3D(#12, #13, #14);
#12=IFCCARTESIANPOINT((0.0, 2.0, 2.0.));
#13=IFCDIRECTION((0.0, 1.0, 0.0.));
#14=IFCDIRECTION((0.0, 0.0, -1.0));
```

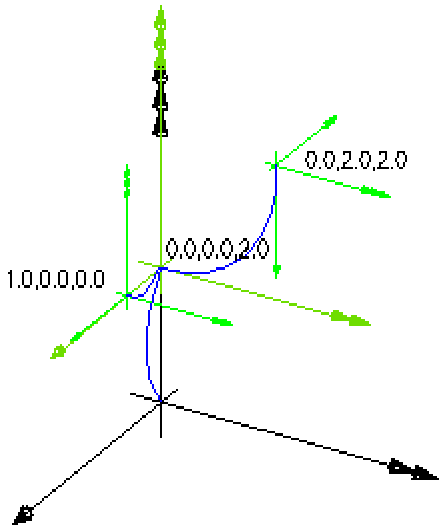

Figure 33 : Example of local relative placement

There are several rules applied to the referencing of parent placements. The referencing has to be a-cyclic and the following shall apply to the subtypes of *IfcProduct*, which have a placement:

- *IfcSite* shall be placed absolutely within the world coordinate system established by the geometric representation context of the project
- *IfcBuilding* shall be placed relative to the local placement of *IfcSite*
- *IfcBuildingStorey* shall be placed relative to the local placement of *IfcBuilding*
- *IfcElement* shall be placed relative:
    - to the local placement of its container (normally to IfcBuildingStorey but possibly also to IfcSite, IfcBuilding,), or
    - to the local placement of the IfcElement to which it is tied by an element relationship (i.e. by one of the following relations IfcRelVoidsElement, IfcRelFillsElement, IfcRelCoversBldgElements, IfcRelAssembles)

2.2.1.2 Concept of local absolute placement

The *IfcLocalPlacement* with the *PlacementRelTo* attribute left unassigned defines the absolute placement of a product in relation to the world coordinate system as established by the assigned geometric representation context of the project. The definition of the axis placement is the same as for the local relative placement (see 2.2.1.1).

Example:

> *Taking a similar example as for the local relative placement, several object placements are used, that are defined absolutely in regard to the world coordinate system. Only the placement objects, IfcLocalPlacement, are shown, but not the subtypes of IfcProduct, which utilizes the placement objects. The IfcGeometricRepresentationContext is given as it establishes the global (or world) coordinate system.*

```
/* Definition of the world coordinate system */
#1=IFCGEOMETRICREPRESENTATIONCONTEXT($, '3Dmodel', 3, 1.0E-05, #2, $);
#2=IFCAXIS2PLACEMENT3D(#3, $, $);
#3=IFCCARTESIANPOINT((0.0, 0.0, 0.0));

/* Definition of the local absolute coordinate systems */
#4=IFCLOCALPLACEMENT($, #7);
#7=IFCAXIS2PLACEMENT3D(#10, $, $);
#10=IFCCARTESIANPOINT((3.8, 1.4, 0.0));

#5=IFCLOCALPLACEMENT($, #8);
#8=IFCAXIS2PLACEMENT3D(#11, #16, #17);
#11=IFCCARTESIANPOINT((1.2, 1.4, 0.0));
#16=IFCDIRECTION((0.0, 0.0, 1.0));
#17=IFCDIRECTION((0.96592, 0.25881, 0.0));

#6=IFCLOCALPLACEMENT($, #9);
#9=IFCAXIS2PLACEMENT3D(#12, #13, #14);
#12=IFCCARTESIANPOINT((2.2., 2.4, 5.0));
#13=IFCDIRECTION((1.0, 0.0, 0.0));
#14=IFCDIRECTION((0.0, 0.0, -1.0));
```

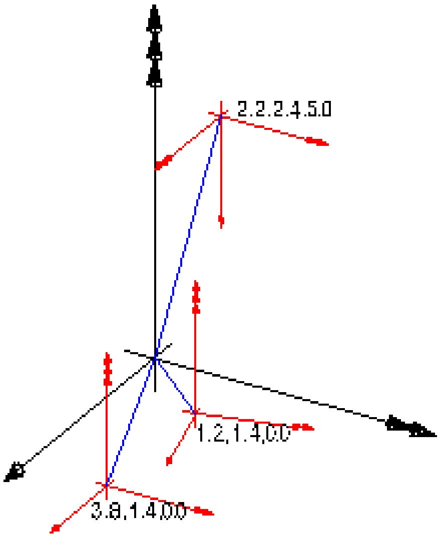

Figure 34 : Example of global absolute placement

2.2.2 Concept of placement relative to grid

The subtype *IfcGridPlacement* allows to definition of an object placement relative to the intersection of two grid lines, *IfcGridAxis*, within the same grid, *IfcGrid* through the *IfcVirtualGridIntersection* relation.

The *IfcVirtualGridIntersection* specifies the object placement of the object coordinate system by pointing to exactly two IfcGridAxis. The intersection point, established by the *PlacementLocation,* defines the *Location* of the object placement (which maybe offset from the exact intersection point). The X-axis orientation of the object placement defaults to the direction of the first intersection axis, if no *PlacementRefDirection* is given, otherwise it is computed by the vector between the *PlacementLocation* and the *PlacementRefDirection*.

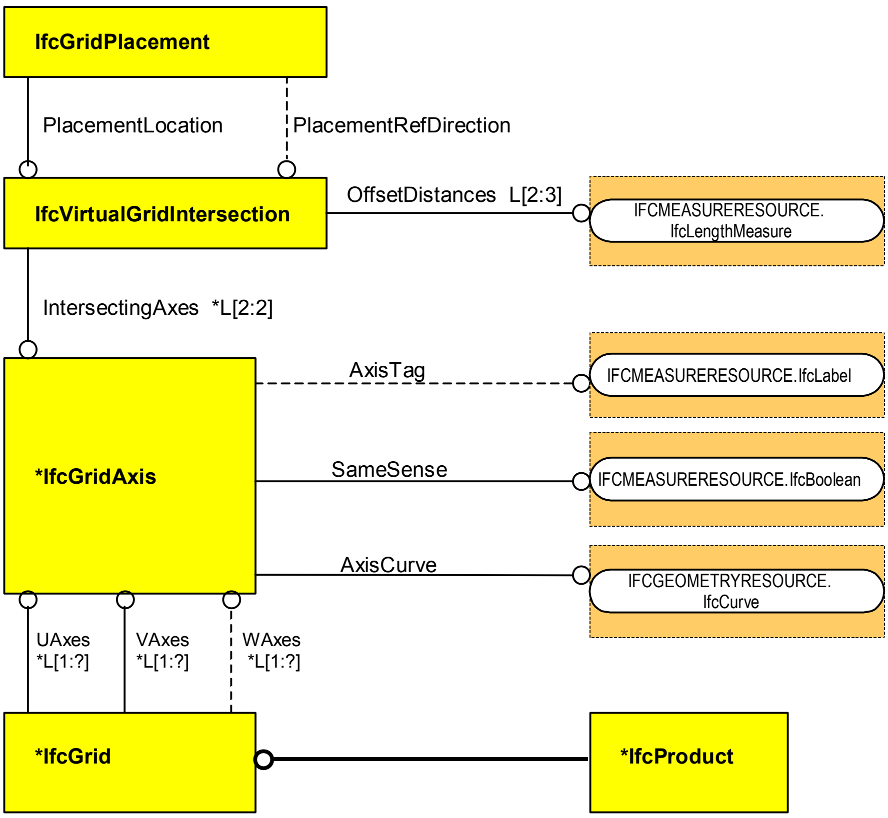

Figure 35 : Definition of placement relative to grid

The functionality supported by the *IfcGridPlacement* thereby includes:

- object placement relative to the intersection of two grid axes
- object placement relative to an offset to the intersection of two grid axes
- object placement relative to the intersection of two grid axes and orientation relative to the direction given by the second pair of grid axes
- object placement relative to an offset to the intersection of two grid axes and orientation relative to the direction given by the second pair of grid axes

Example:

*The placement of an element, e.g. a column (not shown in the *.ifc file), is given relative to the intersection of two grid axes. The placement is offset in the x and y-direction and gets its orientation from the tangent of the first grid axis. The grid axes are defined within the local placement of the grid.*

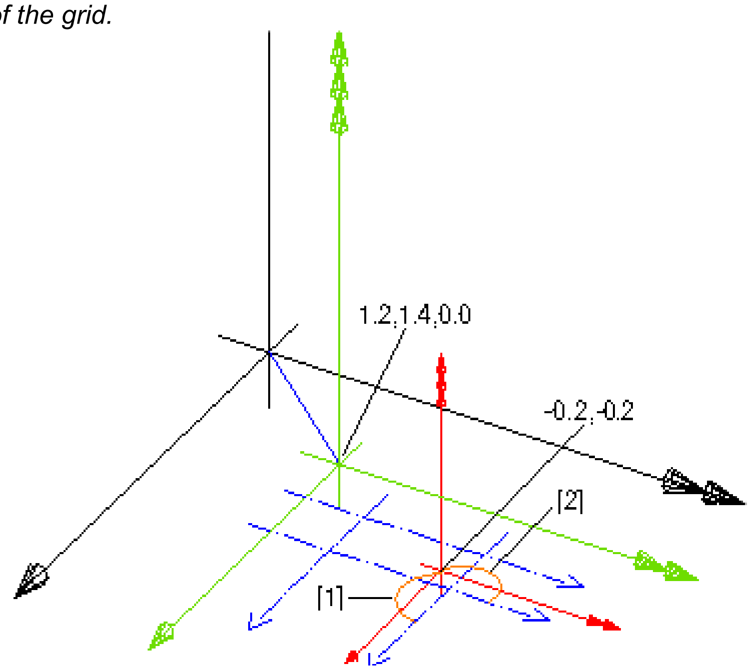

Figure 36 : Example for placement relative to grid

More explanations to the grid placement are later given at the description of the leaf node entity *IfcGrid* elsewhere in the document.

2.3 Concept of multiple product shape representations

Within the same object placement, a product may be represented by a single or multiple shape representations – this functionality is introduced as UoF "multiple shape representations". Each of the individual shape representations may refer to a different representation context and have a different representation name and type. However all geometric representation contexts are required to refer to the same global (or world) coordinate system.

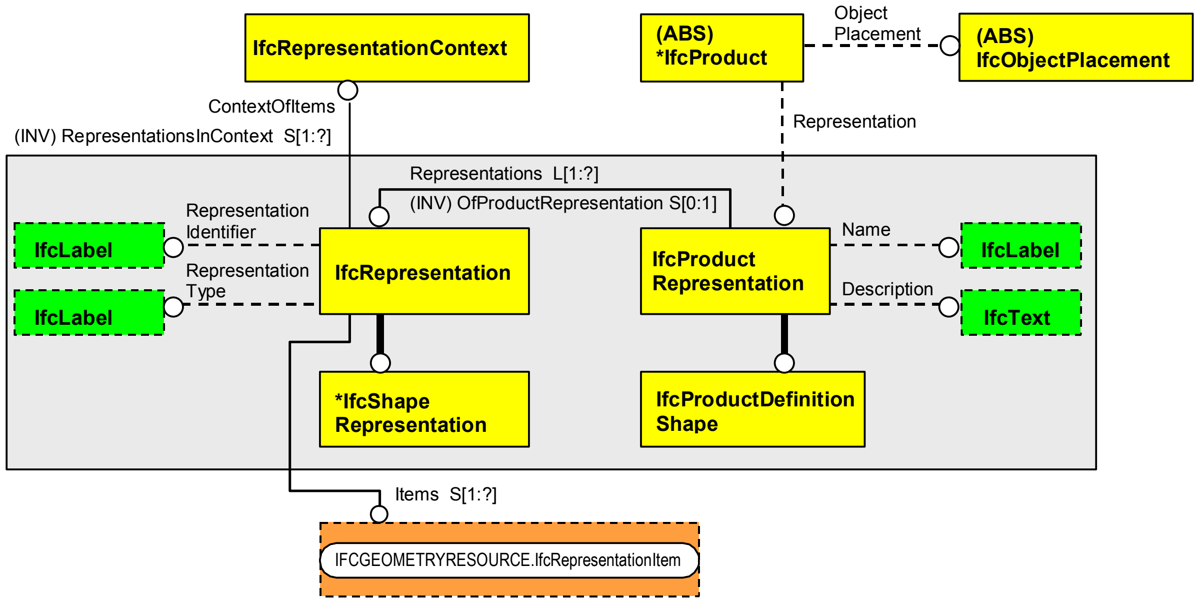

Figure 37 : Definition of (multiple) shape representations

If the product has a product representation, it also has to have an object placement. The object placement provides for the object coordinate system, in which all the geometric representation items of all shape representations are founded. For geometric representations (established by the use of *IfcGeometricRepresentationItem's* as items for single or multiple shape representations) the subtypes *IfcProductDefinitionShape* and *IfcShapeRepresentation* have to be used.

There are two concepts which have to be supported for single and multiple representations of product shape.

- *IfcProductRepresentation* (and subtype *IfcProductDefinitionShape*) as the container of all representations for a product (which are founded in the object placements, as discussed in 2.2)
- *IfcRepresentation* (and subtype *IfcShapeRepresentation*) as the container of all representation items, which are used for a particular representation (within the multiple representations possible) for a product. Each of the *IfcRepresentation*'s would be assigned to a *IfcRepresentationContext* (or subtype *IfcGeometricRepresentationContext*).

2.3.1 Concept of product definition shape

The *IfcProductRepresentation* (or the *IfcProductDefinitionShape* as required for geometric representations) represents the container for all individual, or alternative representations of the same product. It allows for a characterization of the product representation by a *Name* and for the provision of an additional *Description*. Currently there are no recommendations on how to use both attributes. However additional recommendations maybe be given within particular views or implementation agreements.

2.3.2 Concept of shape representation

Each *IfcProductRepresentation* (or subtype *IfcProductDefinitionShape*) has to have at least one *IfcRepresentation* (or subtype *IfcShapeRepresentation*) defined, whereby the subtype *IfcProductDefinitionShape* is required to have at least one *IfcShapeRepresentation* assigned.

Each representation may carry additional classifications provided by the *RepresentationIdentifier* and the *RepresentationType*. The provision of the *RepresentationType* is required for the subtype *IfcShapeRepresentation*. The IFC2x model describes all allowed values for the *RepresentationType* and makes recommendations for the use of the *RepresentationIdentifier*. Therefore both attributes should only be used to carry attribute values are described within the IFC2x specification, mainly within the geometry use sections of the individual subtypes of *IfcElement*.

Example:

The shape representation of a standard wall shall include the wall path and the wall body. It would require a multiple representation of the IfcProductDefinitionShape by assigning two instances of IfcShapeRepresentation. The two attributes RepresentationIdentifier and RepresentationType have the following values:

IfcShapeRepresentation for the wall path:

- RepresentationIdentifier :: "WallAxis"
- RepresentationType :: "Curve2D"

IfcShapeRepresentation for the wall body:

- RepresentationIdentifier :: "WallBody"
- RepresentationType :: "SweptSolid"

The values for *RepresentationIdentifier* and *RepresentationType* have to be specified either within the IFC model specification or within the specific views for implementation. Implementations of the IFC Geometry capabilities should support these conventions.

Example:

*As discussed in the previous example the following picture and the attached *.ifc file cut-out shows the multiple geometric representation of a wall, including the wall axis and the wall basis. It should be noted, that the presentation (i.e. the line color (blue and red) and the line style (continuous and dashed)) is not currently in scope of the IFC2x based exchange. The display of the wall is deter rious software systems, participating in the*

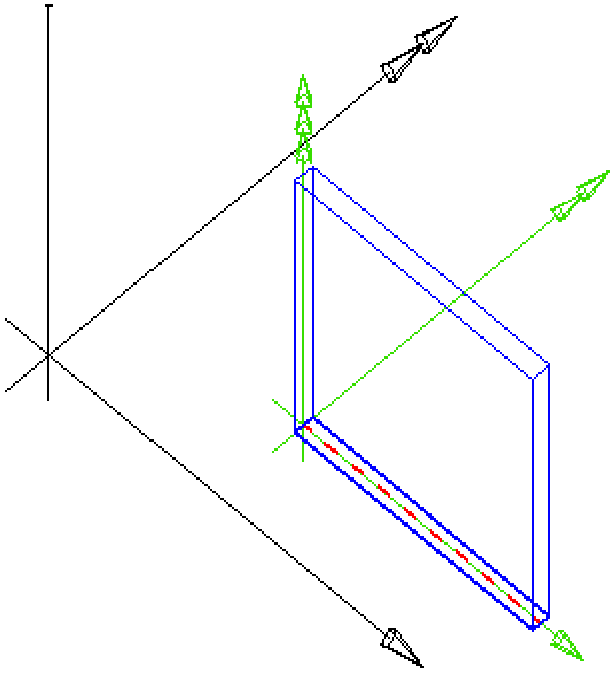

Figure 38 : Example of multiple geometric representation of wall

```
#1=IFCWALLSTANDARDCASE('abcdefghijklmnopqrst01', #2, $, $, $, #3, #4, $);

/* UoF local absolute placement */
#3=IFCLOCALPLACEMENT($, #10);
#10=IFCAXIS2PLACEMENT3D(#16, $, $);
#16=IFCCARTESIANPOINT((2.000000E+00, 1.000000E+00, 0.000000E+00));

/* UoF representation context */
#12=IFCGEOMETRICREPRESENTATIONCONTEXT($, $, 3, $, #14, $);
#14=IFCAXIS2PLACEMENT3D(#15, $, $);
#15=IFCCARTESIANPOINT((0.000000E+00, 0.000000E+00, 0.000000E+00));

/* UoF multiple geometric representations */
#4=IFCPRODUCTDEFINITIONSHAPE($, $, (#11, #13));
```

```
/* first geometric representation for the wall axis */
#11=IFCSHAPEREPRESENTATION(#12, 'WallAxis', 'Curve2D', (#18));
#18=IFCTRIMMEDCURVE(#19,(#20),(#21),.T.,.CARTESIAN.);
#19=IFCLINE(#30, #31);
#30=IFCCARTESIANPOINT((0.0E+00, 0.0E+00));
#31=IFCVECTOR(#32,2.8E+00);
#32=IFCDIRECTION((1.0E+00, 0.0E+00));
#20=IFCCARTESIANPOINT((0.0E+00, 0.0E+00));
#21=IFCCARTESIANPOINT((2.80E+00, 0.0E+00));

/* second geometric representation for the wall body */
#13=IFCSHAPEREPRESENTATION(#12, 'WallBody', 'SweptSolid', (#22));
#22=IFCEXTRUDEDAREASOLID(#23, #26, #29, 2.80E+00);
#26=IFCAXIS2PLACEMENT3D(#28, $, $);
#28=IFCCARTESIANPOINT((0.0E+00, 0.0E+00, 0.0E+00));
#29=IFCDIRECTION((0.0E+00, 0.0E+00, 1.0E+00));
#23=IFCARBITRARYCLOSEDPROFILEDEF(.AREA., $, #40);
#40=IFCPOLYLINE((#41,#42,#43,#44,#41));
#41=IFCCARTESIANPOINT((0.0E+00, 1.0E-01));
#42=IFCCARTESIANPOINT((2.8E+00, 1.0E-01));
#43=IFCCARTESIANPOINT((2.8E+00, -1.0E-01));
#44=IFCCARTESIANPOINT((0.0E+00, -1.0E-01));
```

2.4 Concept of multiple geometric contexts

Each *IfcRepresentation* (and the subtype *IfcShapeRepresentation*) has to be founded within a representation context, given by the UoF "Representation Context", established by the *IfcProject* (see 1.5.6), referencing a single or multiple instances of *IfcRepresentationContext* (or the subtype *IfcGeometricRepresentationContext*).

An IFC project has to have at least one representation context, but may have several contexts. Each context maybe identified by two attributes, *ContextIdentifier* and *ContextType*. Currently there are no recommendations on how to use both attributes. However additional recommendations maybe be given within particular views or implementation agreements.

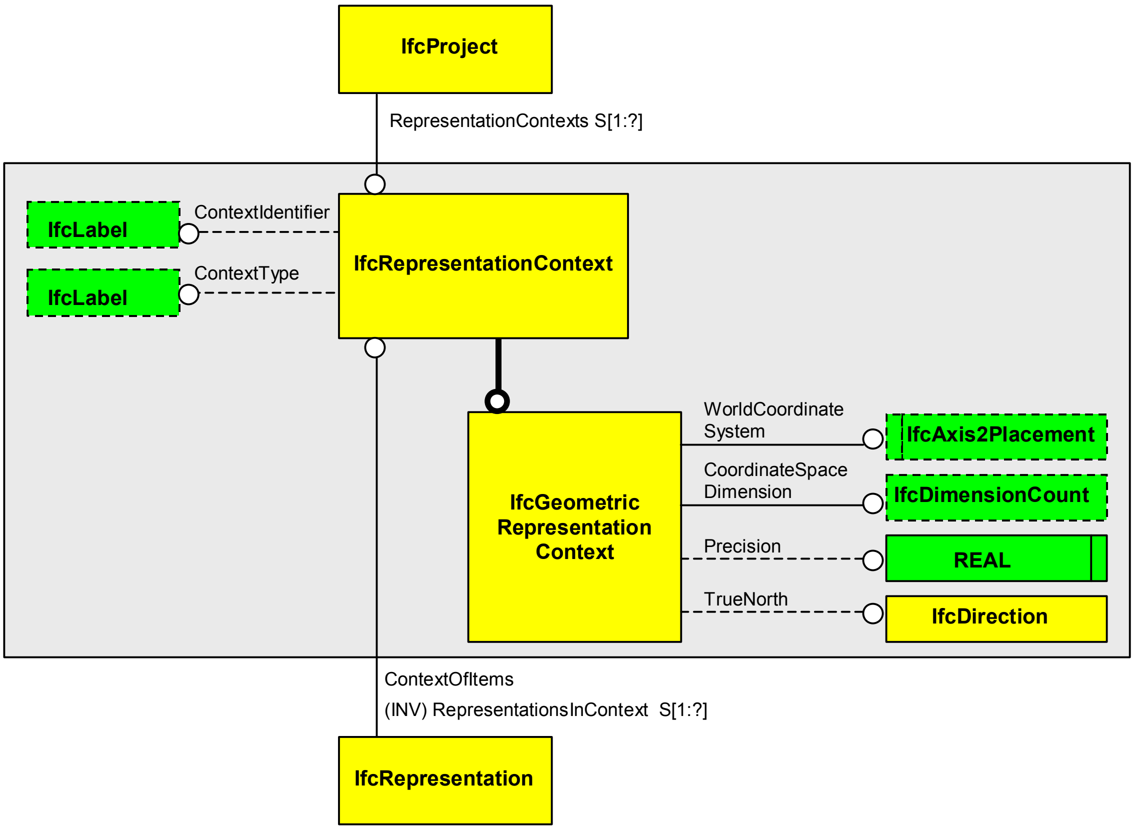

Figure 39 : Definition of (multiple) representation contexts

In case of geometric representations of a product definition shape (as described in 2.3), the subtype *IfcGeometricRepresentationContext* has to be used. In addition to the *ContextIdentifier* and the *ContextType*, the subtype provides two mandatory attributes:

- *WorldCoordinateSystem* : Establishment of the world coordinate system for the representation context used by the project. It is used directly for the UoF "local absolute placement" (see 2.2.1.2), where the object coordinate system in directly placed within the WCS, as established by the geometric representation context. In case of the UoF "local relative placement" (see 2.2.1.1), the uppermost relative placement must relate to the WCS, as established by the geometric representation context1.
- *CoordinateSpaceDimension* : The integer dimension count of the coordinate space used in the geometric representation context.

The two optional attributes *Precision* and *TrueNorth* may give additional information about the context.

- *Precision* : Value of the model precision for geometric models. It is a double value (REAL), typically in 1E-5 to 1E-8 range, that indicates the tolerance under which two given points are still assumed to be identical. The value can be used e.g. to sets the maximum distance from an edge curve to the underlying face surface in brep models. This information is needed for the reliable exchange of B-rep shape models, and should be provided, if shape representations of representation type "Brep" are used.
- *TrueNorth* : Direction of the true north relative to the world coordinate system as established by the representation context. This information maybe provided for the benefit of thermal analysis applications.

If there are multiple geometric representation contexts used the attributes the *ContextIdentifier* and *ContextType* should be used to distinguish the contexts.

2.5 Concept of various shape representation types

Each of the *IfcRepresentation*, and particularly each of the *IfcShapeRepresentation*, are determined by the representation items, which are included within the *Items* attribute. There are three different ways to describe the items of an representation:

- by its (direct) geometric representation
- by its (direct) topological representation
- by its mapped representation, referring to a Cartesian transformation of a source representation

For the representation of shape, topological representation items are not used directly, but only through geometric representation items. Therefore only two UoF, the various forms of direct geometric representation and the mapped representation, are discussed here.

Example:

> *A B-rep of a shape uses the geometric representation item, such as the IfcFacetedBrep, which refers to one or several intermediate topological representation items, such as IfcClosedShell and IfcFace, before being geometrically established by IfcPolyLoop.*

[1] If there are multiple geometric representation contexts within a project, all are required to refer to the same world co-ordinate system. In case of 2D and 3D geometric co-ordination contexts, the XY-plane has to be identical.

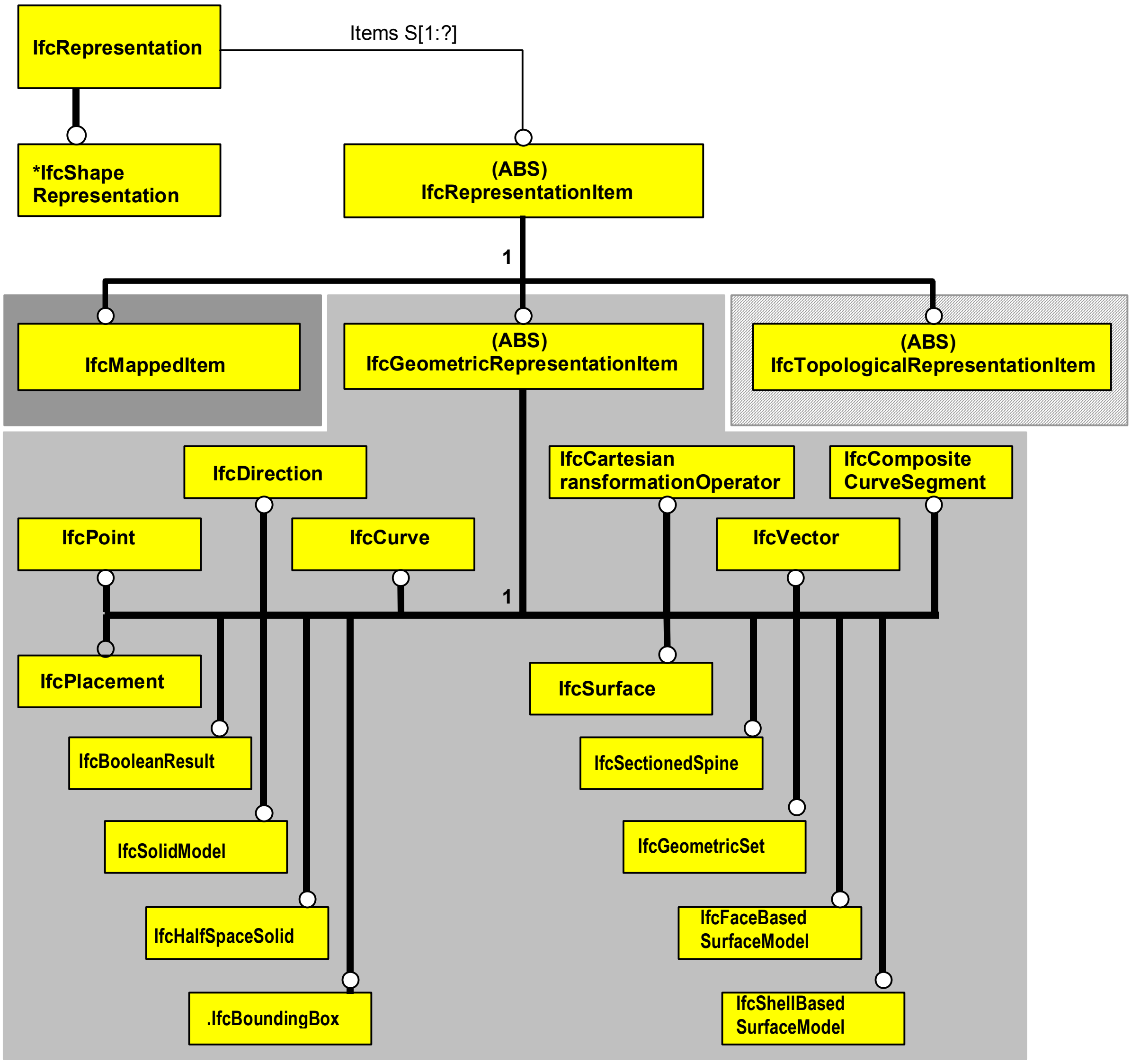

Figure 40 : Concept of shape representations

2.5.1 Concept of direct geometric representation

The attribute *Items* of *IfcRepresentation* (used also by *IfcShapeRepresentation*) allows to assign one or several instances of *IfcRepresentationItem* to describe the representation. For providing the concept of "Direct geometric representation", these instances are required to be all subtypes of *IfcGeometricRepresentationItem*. A further subdivision of the concept of "Direct geometric representation" is established by the various forms of the *RepresentationType* (given as attribute of *IfcRepresentation*).

The following representation types are distinguished:

- **Curve2D** – allows the use of 2D curves for the representation of line based representations, normally used to represent a path;
- **GeometricSet** – allows the use of points, curves, surfaces (both as 2D and 3D elements) for the shape representation (and it includes swept surfaces (both surfaces of extrusion and revolution) for the 3D geometric representation);
  - **GeometricCurveSet** – further restricts the representation type *GeometricSet* to only include points and curves (both as 2D and 3D elements) for the shape representation;
- **SurfaceModel** – allows the use of face based surface model, which includes shells (both open and closed shells) for the shape representation, surface models are always 3D representations;

- **SolidModel** – allows the use of solids, which includes swept solids, Boolean results and B-rep bodies for the shape representation, solid models are always 3D representations. More specific types are:
  - **Brep** – further restricts the representation type *SolidModel* to only include faceted B-rep with and without voids. B-rep models require the satisfaction of the stipulated Euler formulas. No other types of manifold boundary representations then faceted B-rep are currently in scope of IFC2x;
  - **SweptSolid** – further restricts the representation type *SolidModel* to only include swept area solids, including both swept solid by extrusion and swept solid by revolution;
  - **CSG** – further restricts the representation type *SolidModel* to only include Boolean results of operations between solid models, half spaces and other Boolean results. Operations include union, intersection and difference;
    - **Clipping** – further restricts the representation type *CSG* to only include Boolean results resulting in the clipping of an solid and thus restricting the Boolean operation to difference and the second operand to half space solid.
- **BoundingBox** – as an additional representation type it allows for a simplistic 3D representation given by a bounding box. It may be added as an additional representation to enable the display of the shape by application not being able to handle complex shape;
- **SectionedSpine** – as an additional representation type it allows for a cross section based representation of a spine curve and planar cross sections. It can represent a surface or a solid swept along any path (composed of straight lines and arcs) but the interpolations of the form between the cross sections is not defined. The *SectionedSpine* representation may be used within engineering applications for complex duct and pipe shapes (as an example).

2.5.1.1 Concept of Curve2D representation

The representation type *Curve2D* is used to provide a geometric representation of a path based on a bounded curve with a parametric range. It may only be used to provide 2D representation of a linear form (the path).

The *IfcBoundedCurve* should be used to provide the only item of *IfcShapeRepresentation.Items*. For multi-segmented path, the *IfcCompositeCurve* shall be used. The *Curve2D* representation is used, e.g. to provide the wall path information.

See example based on Figure 38 within Section 2.3.2 for an example of how to use *Curve2D* representations.

2.5.1.2 Concept of Geometric Set representation

The representation type *GeometricSet* is used to provide a geometric representation based on a collection of points, lines and surfaces, where no topological structure is available. It may be used to provide either a 2D or a 3D representation of the form, where the items establishing the geometric set should be either all 2D or all 3D but not a mixture of both.

The *IfcGeometricSet* should be used to provide the only item of *IfcShapeRepresentation.Items*. An *IfcGeometricSet* contains one or several *Elements*, which are either subtypes of *IfcPoint*, *IfcCurve* or *IfcSurface*.

The representation type of *GeometricSet* is used, e.g., for the 2D representation of elements, such as furniture, equipment, distribution and building elements, or for special representations, such as the footprint representation of spaces. It also includes general 3D representations, where no solid model is given, and no topological information is available.

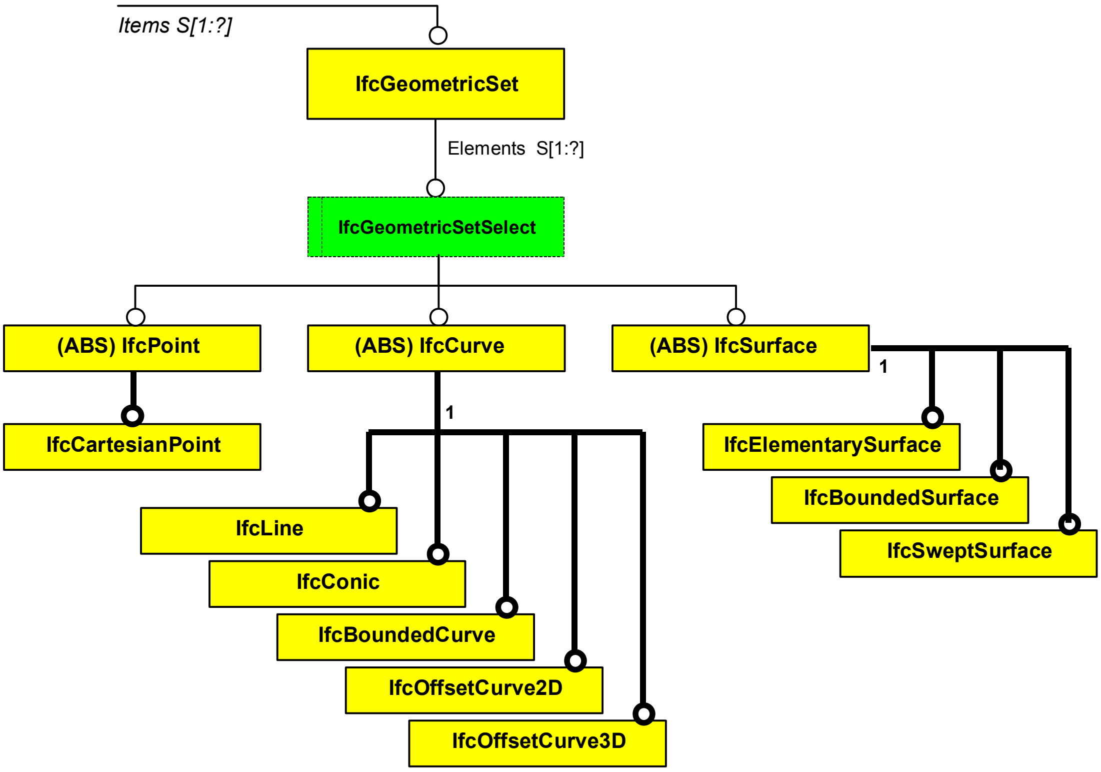

Figure 41 : Definition of Geometric Set representations

The use of geometric set representation for defining a 3D shape (without topological and solid information) includes the swept surface and the rectangular trimmed surface representation (mainly used for revolved surfaces with angle <> 360').

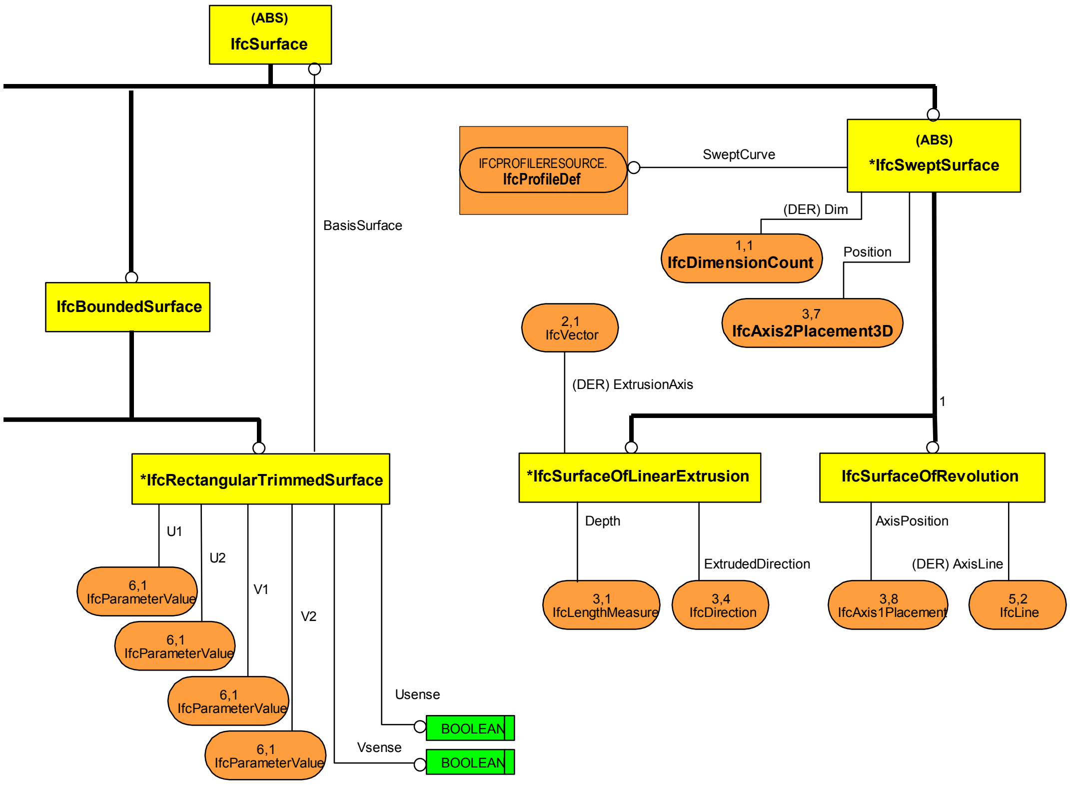

Figure 42 : Definition of surface representations within geometric set

Example:

The geometric representation of the "thin metal" circular duct (which should be given by a surface, not by a solid model) is given by extruding the profile, IfcCircleProfileDef, along a depth. The circle profile definition is a parametric profile, which is mapped into the XY plane of the position coordinate system of the swept surface and extruded (here perpendicular) to the XY plane by the depth.

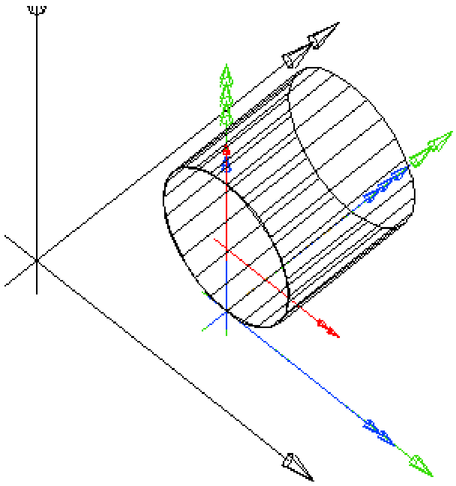

Figure 43 : Example of Geometric Set representation of a duct (using swept surface)

```
#1=IFCFLOWSEGMENT('abcdefghijklmnopqrst12', #2, $, $, $, #3, #4, $, .FLUIDFLOW.,
   .DUCTSEGMENT.);
#2=IFCOWNERHISTORY(#6, #7, .READWRITE., .NOCHANGE., $, $, $, 978921854);

/* UoF local absolute placement */
#3=IFCLOCALPLACEMENT($, #10);
#10=IFCAXIS2PLACEMENT3D(#16, $, $);
#16=IFCCARTESIANPOINT((2.000000E+00, 1.000000E+00, 0.000000E+00));

/* UoF representation context */
#12=IFCGEOMETRICREPRESENTATIONCONTEXT($, $, 3, $, #14, $);
#14=IFCAXIS2PLACEMENT3D(#15, $, $);
#15=IFCCARTESIANPOINT((0.000000E+00, 0.000000E+00, 0.000000E+00));

/* geometric set representation for the duct segment */
#4=IFCPRODUCTDEFINITIONSHAPE($, $, (#11));
#11=IFCSHAPEREPRESENTATION(#12, '', 'GeometricSet', (#13));
#12=IFCGEOMETRICREPRESENTATIONCONTEXT($, $, 3, $, #14, $);
#14=IFCAXIS2PLACEMENT3D(#15, $, $);
#15=IFCCARTESIANPOINT((0.000000E+00, 0.000000E+00, 0.000000E+00));
#16=IFCCARTESIANPOINT((2.000000E+00, 1.000000E+00, 0.000000E+00));

/* use of swept surface for the geometric set */
#13=IFCGEOMETRICSET((#17));
#17=IFCSURFACEOFLINEAREXTRUSION(#18, #21, #25, 2.000000E+00);
#18=IFCCIRCLEPROFILEDEF(.CURVE., $, #19, 1.000000E+00);
#19=IFCAXIS2PLACEMENT2D(#20, $);
#20=IFCCARTESIANPOINT((1.000000E+00, 0.000000E+00));
#21=IFCAXIS2PLACEMENT3D(#22, #23, #24);
#22=IFCCARTESIANPOINT((0.000000E+00, 0.000000E+00, 0.000000E+00));
#23=IFCDIRECTION((0.000000E+00, 1.000000E+00, 0.000000E+00));
#24=IFCDIRECTION((0.000000E+00, 0.000000E+00, 1.000000E+00));
#25=IFCDIRECTION((0.000000E+00, 0.000000E+00, 1.000000E+00));
```

2.5.1.2.1 Concept of Geometric Curve Set representation

The representation type *GeometricCurveSet* is used to provide a geometric representation based on a collection of points and lines, where no topological structure is available. It may be used to provide either a 2D or a 3D representation of the form, where the items establishing the geometric set should be either all 2D or all 3D but not a mixture of both.

The representation type *GeometrcCurveSet* therefore further restricts the type *GeometricSet*, its use is foreseen in future, if further distinctions of geometric sets are required by implementations. Currently the use of geometric curve set is proposed for all cases of point and line representations, that provides 2D representations of elements.

Example:

> *In many cases simple 2D representations of furniture and equipment are sufficient, or may be given in addition to the 3D surface or solid model representations. In this example, the shape of a furniture of type 'Chair' is represented as a 2D geometric set, established within the XY plane of the object coordinate system. The geometric set contains a set of line elements, provided as IfcPolyline's.*

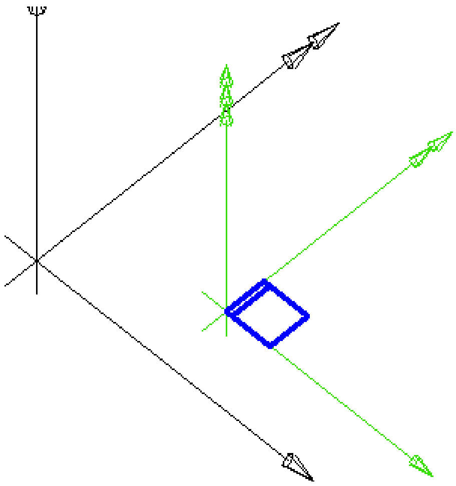

Figure 44 : Example of Geometric Curve Set representation of furniture

```
#1=IFCFURNITURE('abcdefghijklmnopqrst02', #2, $, $, $, #3, #4, $);
#2=IFCOWNERHISTORY(#6, #7, .READWRITE., .NOCHANGE., $, $, $, 978921854);

/* UoF local absolute placement */
#3=IFCLOCALPLACEMENT($, #10);
#10=IFCAXIS2PLACEMENT3D(#16, $, $);
#16=IFCCARTESIANPOINT((2.000000E+00, 1.000000E+00, 0.000000E+00));

/* UoF representation context */
#12=IFCGEOMETRICREPRESENTATIONCONTEXT($, $, 3, $, #14, $);
#14=IFCAXIS2PLACEMENT3D(#15, $, $);
#15=IFCCARTESIANPOINT((0.000000E+00, 0.000000E+00, 0.000000E+00));

/* geometric set representation for the furniture */
#4=IFCPRODUCTDEFINITIONSHAPE($, $, (#11));
#11=IFCSHAPEREPRESENTATION(#12, '', 'GeometricCurveSet', (#18));
#18=IFCGEOMETRICSET((#19, #24));
#19=IFCPOLYLINE((#20, #21, #22, #23, #20));
#20=IFCCARTESIANPOINT((0.000000E+00, 0.000000E+00, 0.000000E+00));
#21=IFCCARTESIANPOINT((0.700000E+00, 0.000000E+00, 0.000000E+00));
#22=IFCCARTESIANPOINT((0.700000E+00, 0.600000E+00, 0.000000E+00));
#23=IFCCARTESIANPOINT((0.000000E+00, 0.600000E+00, 0.000000E+00));
#24=IFCPOLYLINE((#25, #26));
#25=IFCCARTESIANPOINT((0.100000E+00, 0.000000E+00, 0.000000E+00));
#26=IFCCARTESIANPOINT((0.100000E+00, 0.600000E+00, 0.000000E+00));
```

2.5.1.3 Concept of Surface Model representation

The representation type *SurfaceModel* is used to provide a geometric representation based on a collection of faces, provided by face bounds based on poly loops, where a topological structure is available. It may be used to provide a 3D representation of the form. It is normally used to describe the (explicit) 3D form of an object, where no volume information is available or desired.

> *NOTE: Often the rigid solid B-rep representation is not applicable, e.g. for "thin metal" structures, as cupper plates, etc. where an open shell is more applicable to describe the form, or the B-rep representation can not be satisfied by applications without a sophisticated geometry engine. In these cases the SurfaceModel may provide an alternative to exchange the 3D explicit shape.*

The representation type *SurfaceModel* is used, e.g., for the shape representation of distribution (flow) elements, metal plates and other elements.

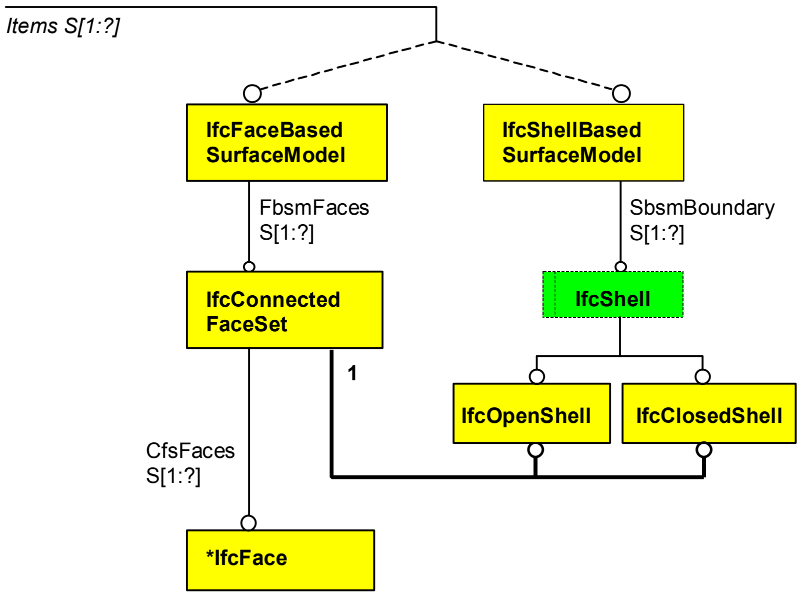

Figure 45 : Definition of Surface Model representations

NOTE: Topological representation items, such as IfcOpenShell, IfcClosedShell, and IfcConnectedFaceSet are not allowed to be immediately referenced by the Items attribute of IfcShapeRepresentation. It is required, that an intermediate geometric representation item, either IfcFaceBasedSurfaceModel or IfcShellBasedSurfaceModel, has to be used.

Example:

Often the use of surface models are more appropriate for building service equipment, although it would not allow clash detection based on the volume of elements. Surface models my also be used for terrain models. The example shows a simple surface model of a duct, given by the IfcDistributionFlowSegment.

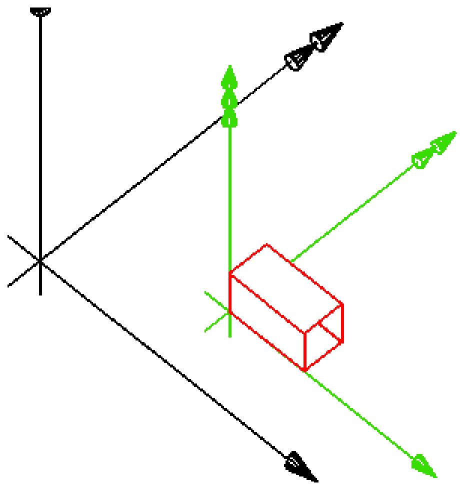

Figure 46 : Example of Surface Model representation of a piece of duct

NOTE: Within the example, a face based surface model had been used, where also a shell based surface model (based on an open shell) would have been available. The decision was arbitrary for the example.

```
#1=IFCFLOWSEGMENT('abcdefghijklmnopqrst02', #2, $, $, $, #3, #4, $,
    .FLUIDFLOW., .DUCTSEGMENT.);
#2=IFCOWNERHISTORY(#6, #7, .READWRITE., .NOCHANGE., $, $, $, 978921854);
```

```
/* UoF local absolute placement */
#3=IFCLOCALPLACEMENT($, #10);
#10=IFCAXIS2PLACEMENT3D(#16, $, $);
#16=IFCCARTESIANPOINT((2.000000E+00, 1.000000E+00, 0.000000E+00));

/* UoF representation context */
#12=IFCGEOMETRICREPRESENTATIONCONTEXT($, $, 3, $, #14, $);
#14=IFCAXIS2PLACEMENT3D(#15, $, $);
#15=IFCCARTESIANPOINT((0.000000E+00, 0.000000E+00, 0.000000E+00));

/* surface model representation for the duct */
#4=IFCPRODUCTDEFINITIONSHAPE($, $, (#11));
#11=IFCSHAPEREPRESENTATION(#12, '', 'SurfaceModel', (#18));
#18=IFCFACEBASEDSURFACEMODEL((#19));
#19=IFCCONNECTEDFACESET((#23, #21, #22, #20));

#20=IFCFACE((#24));
#21=IFCFACE((#25));
#22=IFCFACE((#26));
#23=IFCFACE((#27));
#24=IFCFACEOUTERBOUND(#28, .T.);
#25=IFCFACEOUTERBOUND(#29, .T.);
#26=IFCFACEOUTERBOUND(#30, .T.);
#27=IFCFACEOUTERBOUND(#31, .T.);
#28=IFCPOLYLOOP((#40, #41, #42, #43));
#29=IFCPOLYLOOP((#50, #51, #52, #53));
#30=IFCPOLYLOOP((#60, #61, #62, #63));
#31=IFCPOLYLOOP((#70, #71, #72, #73));
#40=IFCCARTESIANPOINT((0.000000E+00, 0.000000E+00, 0.000000E+00));
#41=IFCCARTESIANPOINT((1.200000E+00, 0.000000E+00, 0.000000E+00));
#42=IFCCARTESIANPOINT((1.200000E+00, 0.000000E+00, 6.000000E-01));
#43=IFCCARTESIANPOINT((0.000000E+00, 0.000000E+00, 6.000000E-01));
#50=IFCCARTESIANPOINT((0.000000E+00, 6.000000E-01, 0.000000E+00));
#51=IFCCARTESIANPOINT((0.000000E+00, 6.000000E-01, 6.000000E-01));
#52=IFCCARTESIANPOINT((1.200000E+00, 6.000000E-01, 6.000000E-01));
#53=IFCCARTESIANPOINT((0.000000E+00, 6.000000E-01, 0.000000E+00));
#60=IFCCARTESIANPOINT((0.000000E+00, 0.000000E+00, 6.000000E-01));
#61=IFCCARTESIANPOINT((1.200000E+00, 0.000000E+00, 6.000000E-01));
#62=IFCCARTESIANPOINT((1.200000E+00, 6.000000E-01, 6.000000E-01));
#63=IFCCARTESIANPOINT((0.000000E+00, 6.000000E-01, 6.000000E-01));
#70=IFCCARTESIANPOINT((0.000000E+00, 0.000000E+00, 0.000000E+00));
#71=IFCCARTESIANPOINT((0.000000E+00, 6.000000E-01, 0.000000E+00));
#72=IFCCARTESIANPOINT((1.200000E+00, 6.000000E-01, 0.000000E+00));
#73=IFCCARTESIANPOINT((1.200000E+00, 6.000000E-01, 6.000000E-01));
```

2.5.1.4 Concept of Solid Model representation

A solid model is the most complete representation of the nominal shape of a product. It determines that all points in the interior are connected and that any point in the domain can be classified as being inside, outside or on the boundary of a solid.

Within the IFC2x model there are three different types of solid model representations defined:

- Boundary model representation (restricted to faceted, manifold b-rep with or without voids);
- Swept area solid representation (using either extrusion or revolution);
- Constructive solid geometry representation, represented by their component solids and the sequence of Boolean operations (it maybe further restricted to solely clipping operations).

The three different specializations of the *SolidModel* representation type then require the correct usage of the different subtypes of *IfcSolidModel* to provide the shape representation of the product.

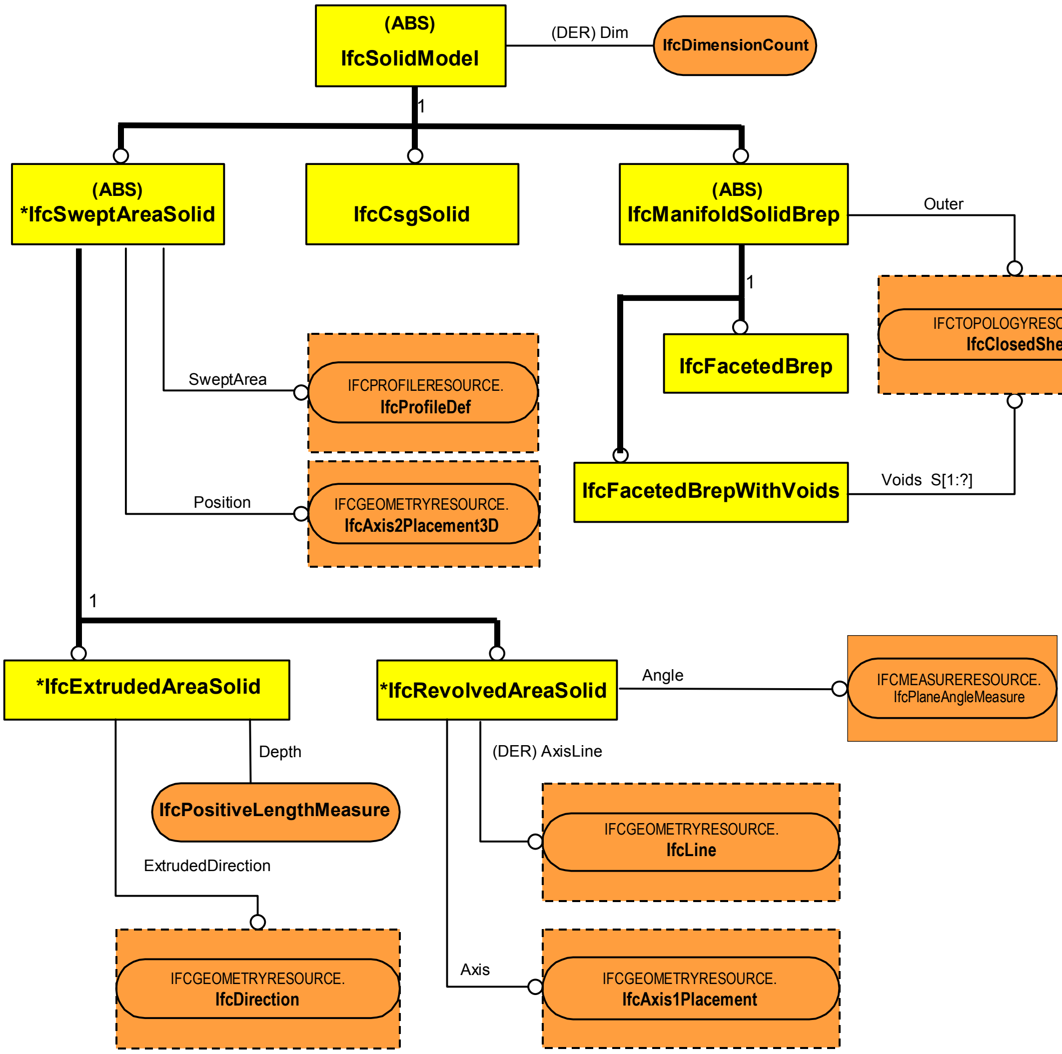

Figure 47 : Definition of IfcSolidModel with subtypes (without CSG)

The *IfcSweptAreaSolid*, when referencing parameterized profile definitions, is normally used as the preferred geometric representation for building elements.

2.5.1.4.1 Concept of B-rep representation

The representation type *Brep* is used to provide a geometric representation based on a collection of faces, provided by face bounds based on poly loops, where a topological structure is available, and the interior (or volume) is defined. Therefore the topological normal of the B-rep at each point of its boundary is the surface normal direction that points away from the solid material. Therefore all faces needs to be (explicitly or implicitly) oriented. All Euler formulas need to the satisfied for the B-rep.

It may be used to provide a 3D representation of the form. It is normally used to describe the (explicit) 3D form of an object, where volume information is available or desired. The exchange of B-rep shape allows for the exchange of (almost) every shape for elements in building & construction.

B-rep representations are restricted to faceted boundary representations only within the IFC2x model. Therefore the only form for face bounds are polyloops, curvatures have to be faceted using approximation techniques.

B-rep representation requires the complete definition of the boundaries of the shape. E.g. all edges of the B-rep are shared by (at least) two faces in the closed shell. In order to determine that the shell is closed and the same edge is referenced twice, the coordinates of the vertices have to be coincident. This can either be achieved by sharing the same instance of the *IfcCartesianPoint* by two or more face bounds (provided through the *IfcPolyLoop* entity), or by using identical copies of the *IfcCartesianPoint*.

In the latter case, the *Precision* attribute of the *IfcGeometricRepresentationContext* determines, which is the maximum difference, under which two points are still be considered to be identical.

Example:

The standard geometric representation of any (non-standard) building element, provided by the IfcBuildingElementProxy, is the Brep representation. The following example shows the Brep shape representation of a proxy with the shape of a box.

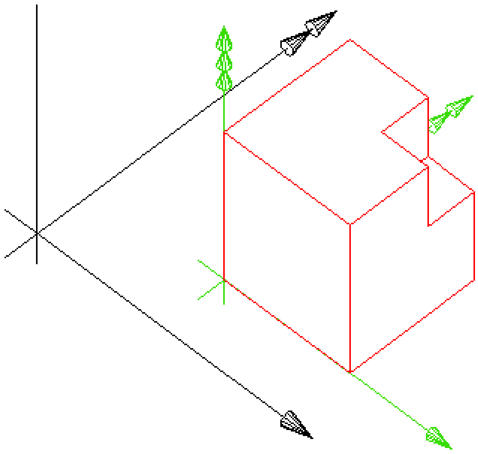

Figure 48 : Example of B-rep Model representation of a proxy

NOTE: The example shows the reference of polyloop vertices by multiple instance reference.[2]

```
#1=IFCBUILDINGELEMENTPROXY('abcdefghijklmnopqrst02', #2, 'Box', $, $, #3, #4, $,$);
#2=IFCOWNERHISTORY(#6, #7, .READWRITE., .NOCHANGE., $, $, $, 978921854);

/* UoF local absolute placement */
#3=IFCLOCALPLACEMENT($, #10);
#10=IFCAXIS2PLACEMENT3D(#16, $, $);
#16=IFCCARTESIANPOINT((2.0, 1.0, 0.0));

/* UoF representation context */
#12=IFCGEOMETRICREPRESENTATIONCONTEXT($, $, 3, 1.0E-06, #14, $);
#14=IFCAXIS2PLACEMENT3D(#15, $, $);
#15=IFCCARTESIANPOINT((0.0, 0.0, 0.0));

/* b-rep model representation for the duct */
#4=IFCPRODUCTDEFINITIONSHAPE($, $, (#11));
#11=IFCSHAPEREPRESENTATION(#12, '', 'Brep', (#174));
#174=IFCFACETEDBREP(#173);
#173=IFCCLOSEDSHELL((#148,#151,#154,#157,#160,#163,#166,#169,#172));

#132=IFCCARTESIANPOINT((0.25,0.25,1.5));
#133=IFCCARTESIANPOINT((1.,0.25,1.5));
#134=IFCCARTESIANPOINT((1.,1.,1.5));
#135=IFCCARTESIANPOINT((0.25,1.,1.5));
#136=IFCCARTESIANPOINT((0.25,0.25,2.5));
#137=IFCCARTESIANPOINT((1.,0.25,2.5));
#138=IFCCARTESIANPOINT((0.25,1.,2.5));
#139=IFCCARTESIANPOINT((-1.,-1.,0.));
#140=IFCCARTESIANPOINT((1.,-1.,0.));
#141=IFCCARTESIANPOINT((1.,1.,0.));
#142=IFCCARTESIANPOINT((-1.,1.,0.));
#143=IFCCARTESIANPOINT((-1.,-1.,2.5));
#144=IFCCARTESIANPOINT((1.,-1.,2.5));
#145=IFCCARTESIANPOINT((-1.,1.,2.5));
#146=IFCPOLYLOOP((#138,#136,#132,#135));
#147=IFCFACEOUTERBOUND(#146,.T.);
#148=IFCFACE((#147));
#149=IFCPOLYLOOP((#136,#137,#133,#132));
#150=IFCFACEOUTERBOUND(#149,.T.);
```

2 In contrary to the previous examples, this example uses "normal" REAL value formats, instead of the notation with exponents in other examples. Both are valid instantiations of the REAL data type and may be used.

```
#151=IFCFACE((#150));
#152=IFCPOLYLOOP((#132,#133,#134,#135));
#153=IFCFACEOUTERBOUND(#152,.T.);
#154=IFCFACE((#153));
#155=IFCPOLYLOOP((#137,#136,#138,#145,#143,#144));
#156=IFCFACEOUTERBOUND(#155,.T.);
#157=IFCFACE((#156));
#158=IFCPOLYLOOP((#138,#135,#134,#141,#142,#145));
#159=IFCFACEOUTERBOUND(#158,.T.);
#160=IFCFACE((#159));
#161=IFCPOLYLOOP((#133,#137,#144,#140,#141,#134));
#162=IFCFACEOUTERBOUND(#161,.T.);
#163=IFCFACE((#162));
#164=IFCPOLYLOOP((#143,#145,#142,#139));
#165=IFCFACEOUTERBOUND(#164,.T.);
#166=IFCFACE((#165));
#167=IFCPOLYLOOP((#144,#143,#139,#140));
#168=IFCFACEOUTERBOUND(#167,.T.);
#169=IFCFACE((#168));
#170=IFCPOLYLOOP((#140,#139,#142,#141));
#171=IFCFACEOUTERBOUND(#170,.T.);
#172=IFCFACE((#171));
```

2.5.1.4.2 Concept of Swept Solid representation

The representation type *SweptSolid* is used to provide a geometric representation based on sweeping a profile (given by a planar bounded area). There are two different types of sweeping operations:

- linear extrusion
- revolution

The position of the swept body depends on the axis placement of the swept area solid, where the XY plane of the placement is used to place the profile.

2.5.1.4.3 Concept of CSG representation

The representation type *CSG* is used to provide a geometric representation based on the CSG model of the represented object. A solid represented as a CSG model is defined by a collection of so-called primitive solids, combined using regularized Boolean operations.

The use of CSG is currently restricted to the Boolean operations on other solid models, as no CSG primitives are included in the IFC2x specification. The provision of CSG representations mainly allows for clipping operations between swept area solids and half space solids.

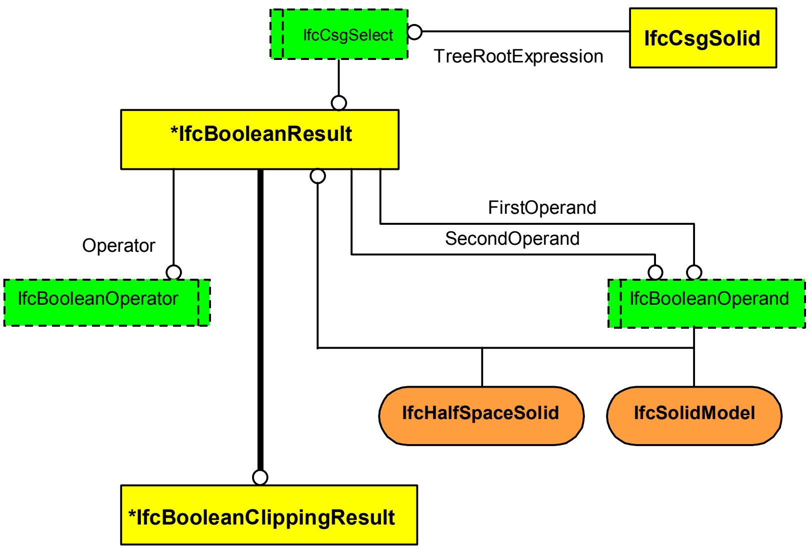

Figure 49 : Definition of Boolean results

NOTE: Current Geometry use definitions do not foresee the use of other Boolean operations, that the clipping (expressed by a Boolean difference), therefore no Boolean unions or intersections are momentarily used.

2.5.1.4.3.1 Concept of Clipping representation

The representation type *Clipping* is used to provide a geometric representation based on the CSG model of the represented object, where the first operand is a solid model, or another Boolean result within the CSG tree and the second operand is a half space solid. The Boolean operator is always DIFFERENCE.

The representation type *Clipping* is normally used to clip part of the solid (given by a swept area solid), whereas several clippings allowed within the CSG tree.

3 Property Definition

The IFC Object Model comprises a set of well defined ways of breaking down information into classes and the structure of information that defines an objects (which is an instance of a class in use). The information structures provide a formal specification of attributes that belong to classes and define how data exchange and sharing using ISO 10303 parts 21 and 22 or other encoding method will be achieved.

However, there are many types of information that users might want to exchange that are not currently included within the IFC Model.

For this purpose the IFC Model provides the Property Definition mechanism (part of which is within the *IfcKernel* schema with the remainder being within the *IfcPropertyResource* schema). Property Definition is a generic mechanism that allows model users and developers to define, connect and use data-driven, expandable properties with objects

Property Definitions can be either:

- type defined and shared among multiple instances of a class, or
- type defined but specific for a single instance of a class, or
- extended definitions that are added by the end users.

3.1 Extended Concept of Property Definition

In addition to the short introduction within section 1.7, the concepts provided by *IfcPropertyDefinition* and its subtypes are further described here. It allows for:

- Relating of an object type, for which a set of properties is defined. This is done though assigning an IfcTypeObject (through the IfcRelDefinesByType relationship) to a single or multiple object occurrences.

  For instance, there may be a class called IfcFan within the statically defined model[3]. However, the different types of fan that may exist (single stage axial, multi-stage axial, centrifugal, propeller etc.) are not in the statically defined model. These may be declared as types of the fan through a type relationship attached to the IfcFan class. Each type of fan that could be defined in IFC is included in an enumeration of fan types, or maybe extended by a type designation captured in the general ObjectType string attribute. The value that these attributes take defines the IfcTypeObject that is assigned.

- Sharing a standard set of values defined in an *IfcPropertySet* (or any other subtype of *IfcPropertySetDefinition*) across multiple instances of that class (through the *IfcRelDefinesByProperties* relationship).

  For instance, a standard range of properties with known values might be defined for the maintenance of centrifugal fans. These properties will be applied to every centrifugal fan by attaching the same instance of IfcPropertySet to all instances of IfcFan through the same 1:N relationship instance IfcRelDefinesByProperties.

- Defining different property values within a private copy of the *IfcPropertySet* for each instance of that class.

  For instance, all centrifugal fans deliver a volume of air against a known resistance to airflow. Although these properties are assigned to every centrifugal fan, the values given to them differ for every instance. There are two options to attach these private properties:

  - If they overlap with properties within the shared property sets, i.e. they redefine a commonly shared property value, the properties should be assigned as overriding properties through the IfcRelOverridesProperties relationship.
  - If they add to the commonly shared properties within the assigned property sets, then they should be added by another instance of IfcRelDefinesByProperties, assigning the specific single property set.

[3] The term ‘statically defined model’ is used to refer to the EXPRESS specification part of the IFC Model. Use of the term ‘dynamic’ or ‘dynamically defined model’ is restricted to those parts of the IFC Model that are not formally defined in EXPRESS or extensions to the IFC Model that are not documented in an IFC release.

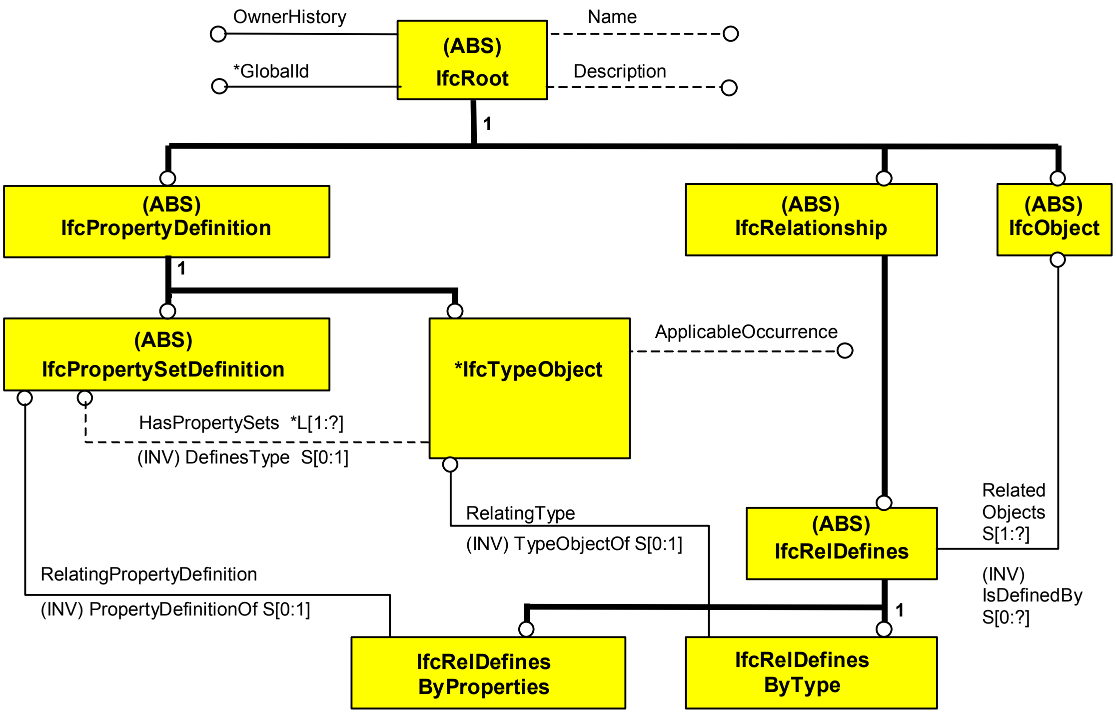

Figure 50 : Definition of Property Definitions

3.2 Extended concept of Property Set Definition

In addition to the short introduction within section 1.7.3, the concepts of *IfcPropertySetDefinition* and its subtypes are further described here.

The *IfcPropertySetDefinition* is an abstract supertype of property sets that can be used within the IFC Model. These include both the dynamically defined property sets declared through the *IfcPropertySet* class and the statically defined property sets such as the *IfcManufactureInformation* class, which represents a subtype of *IfcPropertySetDefinion* defined elsewhere in the IFC2x model. Both types of property set definitions can be distinguished as:

- **Statically defined properties**, they define properties for which a entity definition exist within the IFC model. The semantic meaning of each statically defined property is declared by its entity type and the meaning of the properties is defined by the name of the explicit attribute.
- **Dynamically extendable properties**, they define properties for which the IFC model only provides a kind of "meta model", to be further declared by agreement. This means no entity definition of the properties exists within the IFC model. The declaration is done by assigning a significant string value to the "Name" attribute of the entity as defined in the entity IfcPropertySet and at each IfcProperty, referenced by the property set.

3.2.1 Property Set Definition Attachment

Both, statically and dynamically defined property sets are attached to an object using the relationship class *IfcRelDefinesByProperties*. This allows for the object and the property definition to exist independently and for the *IfcPropertySetDefinition* to be attached to the object when required. An advantage of using the relationship class is that the object does not contain any references to property set definitions if none needed.

Use of the relationship class also allows the property set definition to be assigned to one or many objects. That is, many objects may share a single property set definition with common values if required.

> *For instance, if there are 28 chairs (instances of the IfcFurniture class) that are all exactly the same, they can all share a reference to the same IfcPropertySetDefinition that defines information about the type of upholstery, color etc.*

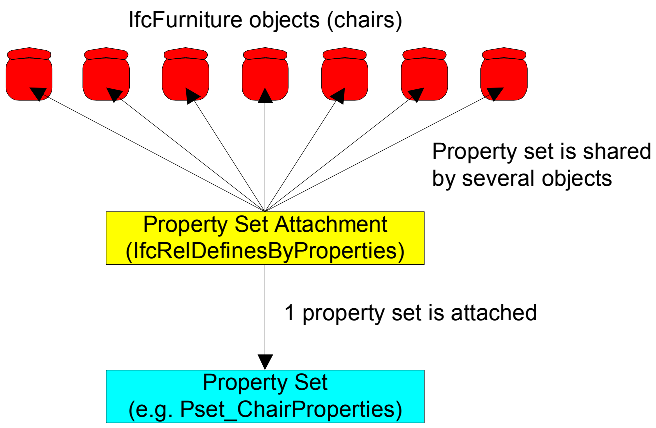

Figure 51 : Example of Property Set Atachment

Both the *IfcObject* and the *IfcPropertySetDefinition* have inverse attributes to *IfcRelDefinesByProperties*. Whilst these cannot be seen in an IFC exchange file formatted according to ISO 10303 Part 21, they can be used by an application. For instance, an object may be defined by many property definition assignments and these can be determined by the inverse attribute.

3.2.2 Overriding of Property Definitions

A property set may be assigned to many objects. However, it may be the case the value of some properties in a subset of the objects may vary whilst others remain constant. Rather than defining and assigning new property sets, the IFC Object Model provides the capability to override those properties whose values change. This is done by the assignment of an overriding property set that contains new values for those that have varied.

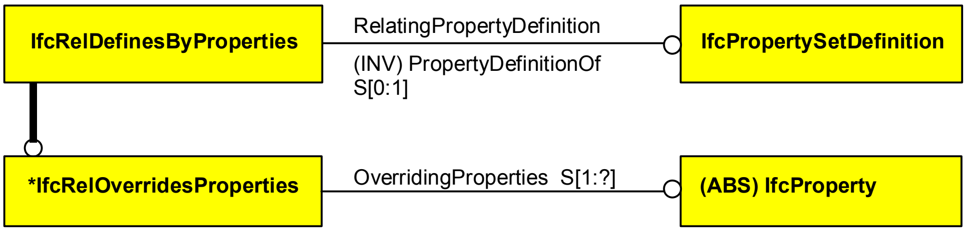

Figure 52 : Definition of IfcRelOverridesProperties

Example:

> *Figure 53 shows a set of objects (here chairs) that are to be assigned a property set that contains properties. Of this set of chairs, there is one chair which has a different value for one property (here color). Therefore the standard value within the shared property set is overridden by the specific value for that individual chair. The value of all other properties in the subset remain unchanged, as do the values of all properties assigned to the other objects.*

Note that there must be a total correspondence between the names of the properties in the set of overriding properties and the names of the properties whose values are to be changed in the base property set. In addition the inherited attribute *RelatingPropertyDefinition* points to the property set which values are overridden.

The pointer to the actual original property set is necessary to avoid redundancies, if the overriding properties may appear (by name) at more than on property set, attached to the object.

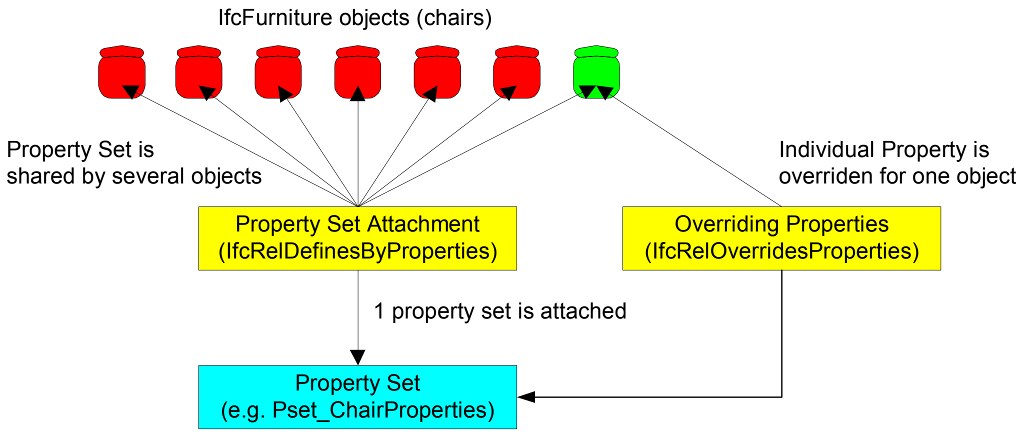

Figure 53 : Example of Overriding Properties

3.3 Extended concept of Type Object

In addition to the short introduction within section 1.7.1, the concepts of *IfcTypeObject* and its subtypes are further described here.

An *IfcTypeObject* provides the means for grouping together a number of property sets that commonly work together. This is in preference to a situation where every property set is individually assigned to an object (although it is still possible to assign property sets individually). An *IfcTypeObject* may act as the container for a list of property sets (property set definitions) where the list is ordered (each property set is in the same relative location for each instance of the *IfcTypeObject*) and there is no duplication of property sets. An inverse relationship between *IfcPropertySetDefinition* and *IfcTypeObject* provides for the possibility of relating the definition to an object type within which it is contained.

- Each *IfcTypeObject* has a name (inherited from *IfcRoot*) that identifies it. This could, for instance, be used in the context of library information as a means of acquiring several property sets at the same time and assigning them to an object via the *IfcTypeObject*.
- The *Description* attribute (also inherited from *IfcRoot*) may be used to provide further, human readable, narrative information about the object type.
- The *ApplicableOccurrence* attribute may be used to constrain the *IfcTypeObject* with regard to the class within the IFC Model to which it can be assigned. This acts in the same manner as for type defined classes in previous releases of the IFC Model.

3.3.1 Type Object Attachment

The *IfcTypeObject* instance is attached to an object using the relationship class *IfcRelDefinesByType*. This allows for the object and the *IfcTypeObject* to exist independently and for the *IfcTypeObject* to be attached to the object when required. An advantage of using the relationship class is that the object does not contain any references to property set definitions if none are needed.

Use of the relationship class also allows the *IfcTypeObject* to be attached to one or many objects. That is, many objects may share a single *IfcTypeObject* with its contained property set definitions if required.

> *For instance, if there are 28 chairs (instances of the IfcFurniture class) that are all exactly the same, and there are multiple property sets that act together within an IfcTypeObject to describe the chair they can all share a reference to the same IfcTypeObject.*

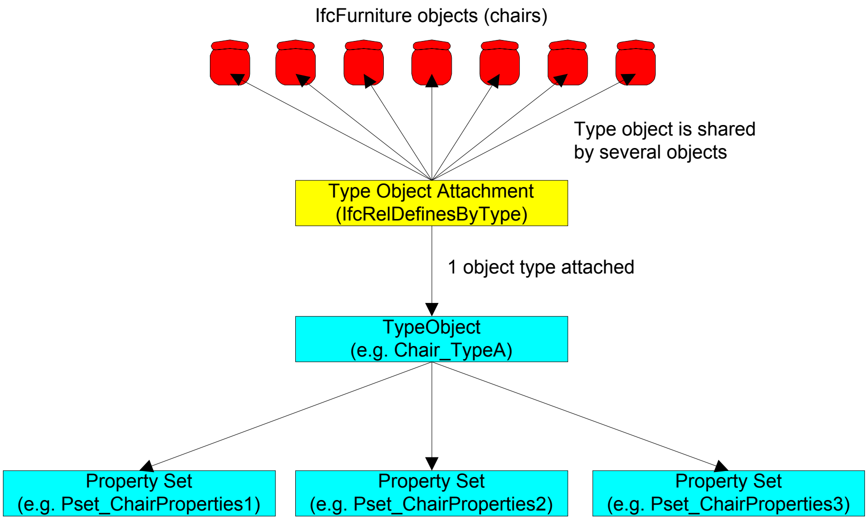

Figure 54 : Example of Type Object Attachment

3.4 Extended concept of Type Product

In addition to the short introduction within section 1.7.2, the concepts of *IfcTypeProduct* are further described here.

The *IfcTypeProduct* class enables a property definition to have one or more shape representations through the *RepresentationMaps* attribute. This attribute has the type *IfcRepresentationMap* that is the class in the *IfcGeometryResource* schema that defines a group of geometrical objects that can act together and be assigned to multiple objects in the manner of a symbol or block in a CAD system.

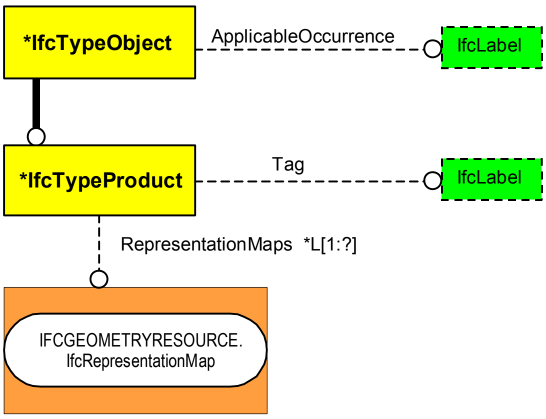

Figure 55 : Definition of IfcTypeProduct

Example:

> *All instances of the chair example, all instances of IfcFurniture, does not only share the same set of properties (by a common IfcTypeObject), but also the same geometric representation. Therefore the subtype IfcTypeProduct is used, which allows the sharing of the same representation maps. A set of representation maps can be seen as a multi-view block, i.e. a block definition, that includes several blocks for different representation views. For example a 2D representation of the chair and a 3D B-rep representation of the chair.*
>
> *Each instance of the IfcFurniture would then have its own local placement (see) and (multiple) shape representations, each holding an IfcMappedItem, which references one map of the shared representation maps.*

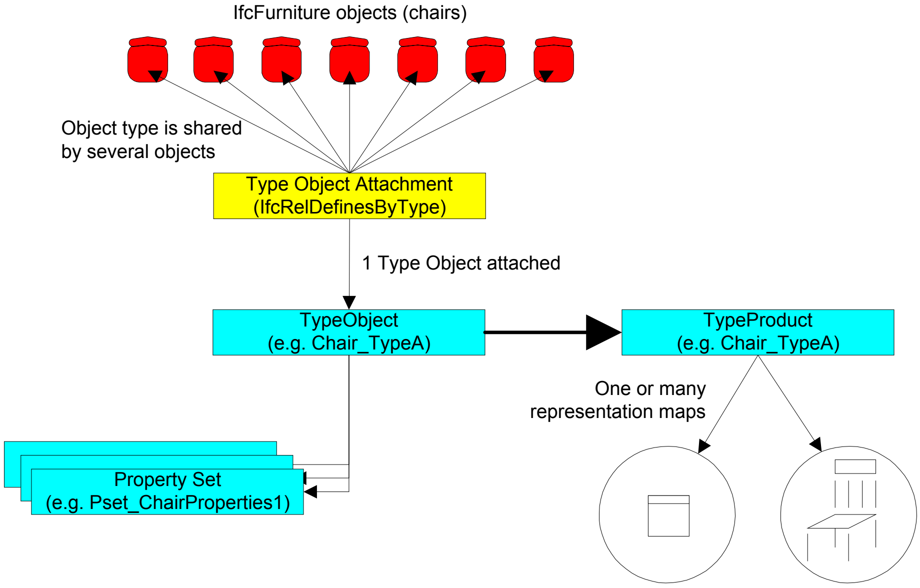

Figure 56 : Example of the IfcTypeProduct Class

3.5 Extended concept of dynamic Property Sets

The *IfcPropertySet* class is a container that holds collections (or sets) of properties. A property set is defined externally to the statically defined IFC Model in the EXPRESS data definition language.

A number of property sets are defined and distributed with the IFC Model. Those property sets do form part of the complete IFC Model (together with the EXPRESS definition), although they are not part of the IFC2x platform.

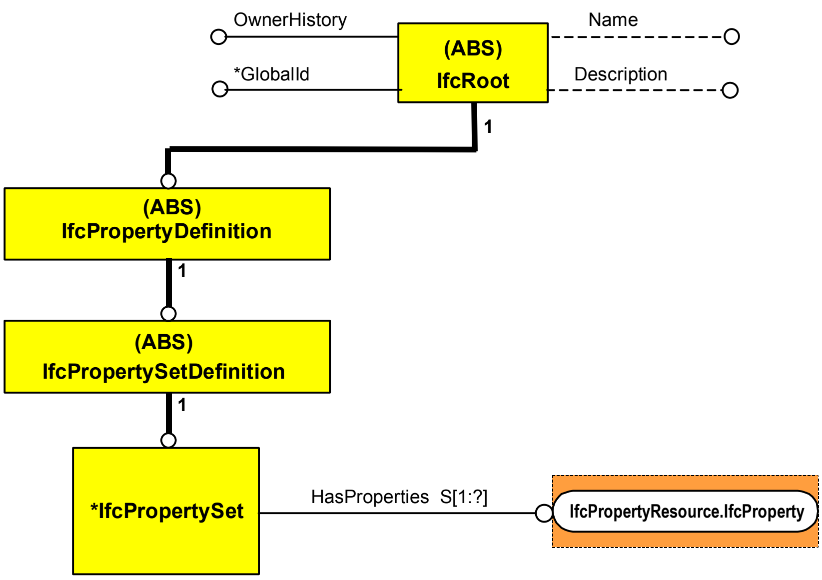

Figure 57 : Definition of IfcPropertySet

The specification of a property set defines the manner in which property information is held within a fully populated IFC exchange file or within an IFC compliant database. It can also be used to define the specification for the transport of a message containing *IfcPropertySet* information.

- An *IfcPropertySet* contains a set of properties. There must be at least one property within a property set.
- An *IfcPropertySet* has a Name (inherited from *IfcRoot*). This is an identifier that is used for recognition. It is a vitally important attribute in the context of defining industry or project standard property sets that may be used by multiple organizations.

- An *IfcPropertySet* may also have a description (inherited from *IfcRoot)* that is a human readable narrative describing some information about the property set.

3.6 Concept of dynamic Property Definitions

The fundamental aspect of the *IfcPropertySet* is that it contains a set of properties. It must contain at least one property and may contain as many as are necessary. Since it contains a SET of properties, the order in which the properties appear is of no significance. Only the names of the individual properties characterize their content and meaning.

A property may be either of the following (each of these types of property is further described below):

- a single value (with or without units),
- an enumeration value (with or without units),
- a bound value (with or without units),
- a range of values (with or without units),
- an object reference,
- a complex property,

The *IfcProperty* class is the common abstraction for all Properties defined within the IFC Model. It is an abstract supertype, meaning that there is never an *IfcProperty* object itself, only objects that are a subtype of *IfcProperty*. Every instance of *IfcProperty* must have a *Name* by which it can be identified. It may have an *Description*, to further describe the meaning of the property.

As use of the dynamic parts of the IFC Model expands, it is intended that a dictionary of standard IFC properties will be defined progressively. For the present, for those properties used in property sets that are published as part of the IFC Model, overlap in naming, definition and usage of units has been eliminated.

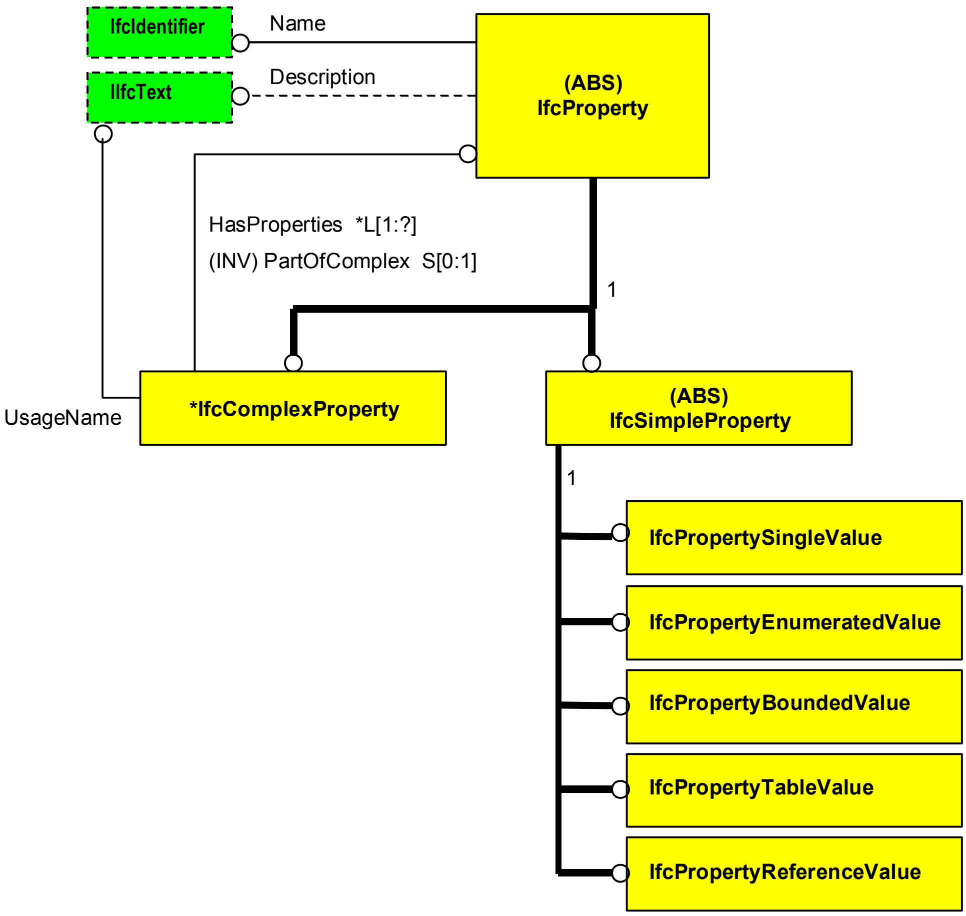

Figure 58 : Definition of IfcProperty

3.6.1 Concept of Property with Single Value

An *IfcPropertySingleValue* is a single property that has either a *name -- value* pair or a *name – value – unit* triplet. Provision of a unit is optional.

If no *Unit* is assigned, the globally defined unit (see 4.2) is used as it relates to the measure type of the *NominalValue* (chosen from the *IfcValue* select type). If a *Unit* is provided, the unit for the nominal value can be defined locally (and it thereby overrides the eventually given globally defined unit). This concept applies to all subtypes of *IfcSimpleProperty*.

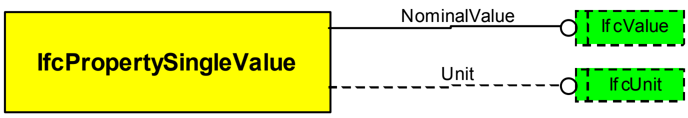

Figure 59 : The IfcPropertySingleValue Class

3.6.2 Concept of Property with Enumerated Value

An *IfcPropertyEnumeratedValue* allows for the (potentially multiple) selection of a property from a predefined list of selections. The actual value is stored by the attribute *EnumerationValues* and is selected from the list that is defined within an *IfcPropertyEnumeration* object.

The *EnumerationValues* allow for either a single choice (if the list only contains a single item) or for multiple choices. The *IfcPropertyEnumeration* instance may be referenced by the *EnumerationReference* attribute – if it is given, a where rule ensures, that the selected values are within the list of the values as specified in the *IfcPropertyEnumeration*.

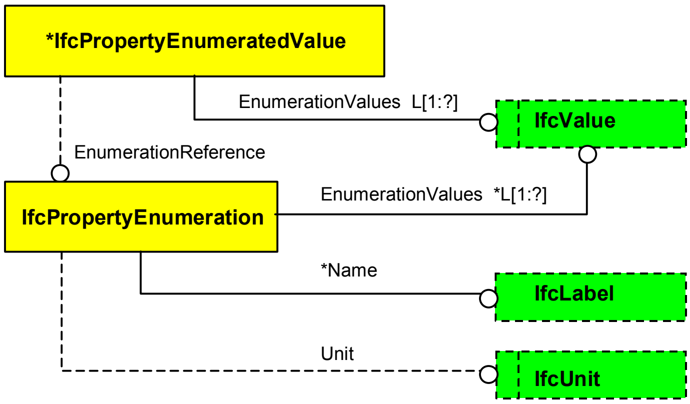

Figure 60 : Definition of IfcPropertyEnumeratedValue

The list of possible selections is defined through the use of the *IfcPropertyEnumeration* class. The actual values from which the selection is made are stored in the attribute *EnumerationValues*. Each *IfcPropertyEnumeration* has a name that must be unique to distinguish it from other instances of *IfcPropertyEnumeration* that may be used.

A unit may be assigned to an *IfcPropertyEnumeration*. This is the unit that each value in the enumeration shares. It is not possible to have values within the enumeration that have different units.

3.6.3 Concept of Property with Bounded Value

An *IfcPropertyBoundedValue* allows for a property whose value can be allowed to vary between an upper limit (the *UpperBoundValue* attribute) and a lower limit (the *LowerBoundValue* attribute).

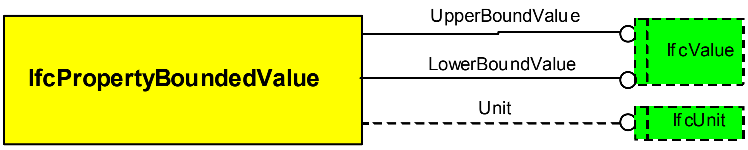

Figure 61 : Definition of IfcPropertyBoundedValue

3.6.4 Concept of Property with Table Value

An *IfcPropertyTableValue* allows for the definition of a range of values where each value stored is dependant on another value. This allows for either values in a two dimensional table to be stored or approximates to the storage of a set of values that may be derived from an expression for x and y where $y = \phi(x)$

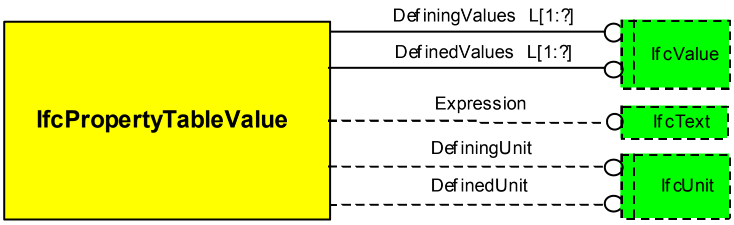

Figure 62 : Definition of IfcPropertyTableValue

Two value ranges are defined namely the *DefiningValues* and the *DefinedValues*. The *DefiningValues* are those upon which the *DefinedValues* are dependent. For example, if an *IfcPropertyTableValue* object was storing values where the expression $y = x^2$ was applicable, the table of values would be:

DefiningValues	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*
DefinedValues	*1*	*4*	*9*	*16*	*25*	*36*	*49*	*64*

The Expression from which the values are derived may also be stored for reference. Using the Expression attribute is for convenience; no operations are carried out on the expression. The Units that are used for both the *DefiningValues* and the *DefinedValues* may also be defined.

3.6.5 Concept of Property with Reference Value

An *IfcPropertyReferenceValue* enables reference to objects whose structure is defined by the static part of the IFC Object Model. This is achieved by defining a relationship to the object being referenced. This is achieved through the *IfcObjectReferenceSelect* that allows selection of the type of object (class) required. An *IfcPropertyReferenceValue* may have a usage name that defines the purpose or usage

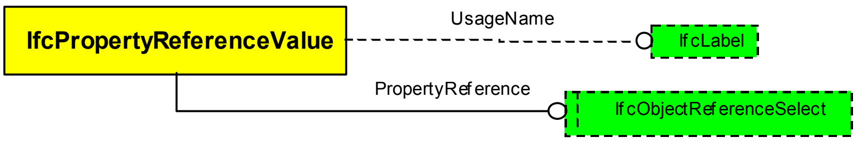

Figure 63 : Definition of IfcPropertyReferenceValue

The *IfcObjectReferenceSelect* defines the types of object (classes) that may be referenced as a property within an IfcPropertySet object. The purpose is to make available the capabilities of the class as though it was a property.

There are a limited number of classes that may be selected through the use of the *IfcObjectReferenceSelect* data type. All of these classes occur within the Resource layer of the IFC Object Model. This restriction conforms to the provisions of the IFC Technical Architecture which requires that an object can only reference a class at the same or a lower layer within the Architecture. Since an *IfcPropertyReferenceValue* is itself a class that exists at the Resource layer, it can therefore only refer to other classes at the Resource layer.

3.6.6 Concept of Complex Property

An *IfcComplexProperty* provides the means for extending the range of properties that can be assigned to an object by defining a mechanism for bringing other properties into named groupings.

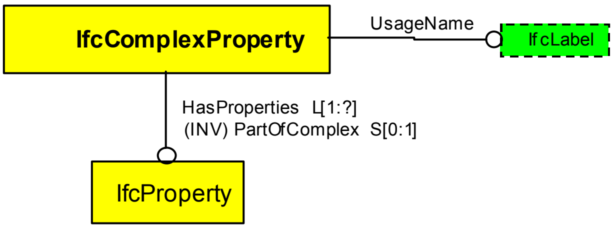

Figure 64 : Definition of IfcComplexProperty

Since any *IfcComplexProperty* can include other properties (being either *IfcSimpleProperty* or *IfcComplexProperty*) they can be assigned in a tree structure by:

- An *IfcPropertySet* contains a set of properties, one or more of which may be of type *IfcComplexProperty*.
- An *IfcComplexProperty* contains a one or more other properties amongst which may be zero, one or more of type *IfcComplexProperty*

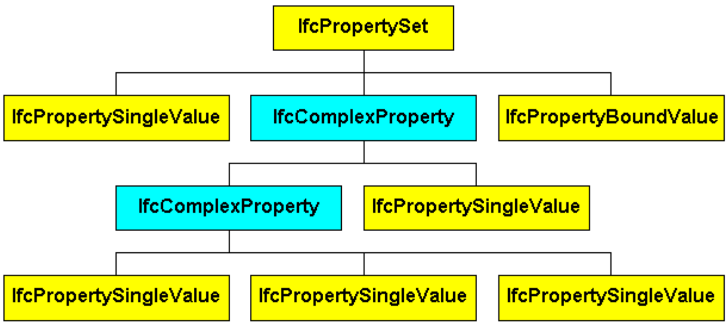

Figure 65 : Example of Nested Complex Properties

An *IfcComplexProperty* has a *UsageName* that describes how the property is to be used within a property set. This is particularly important where there may be more than one complex property of the same type used within a property set. This can happen where there is more than one item of the same type within an object but the usage of each item differs.

Example:

> *Consider two complex properties of the same name that include glazing properties. The Name attribute of the complex property could be Pcomplex_GlazingProperties. The UsageName attribute for one instance of the complex property could be OuterGlazingPlane whilst for the other instance, the UsageName attribute could be InnerGlazingPane. This would distinguish the usage of the IfcComplexProperty for the two glazing panes.*

A rule on the IfcComplexProperty class prevents the assignment of more than one copy of identical complex properties within a property set.

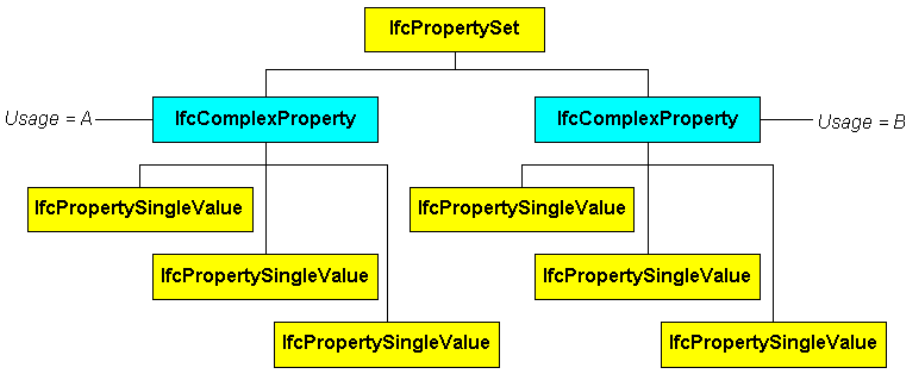

Figure 66 : Example of Complex Properties with Different Usage

4 Unit Definition and Assignment

In the IFC object model there may be a number of ways to express the measure defined types (for measure values) and their units. The purpose of this section and the examples is to promote common understanding how measure types and related units should be used, i.e. instantiated.

The definitions of units and measure types are provided in the *IfcMeasureResource* and had been based on the specification of the measure_schema as defined in ISO 10303-41 "Integrated Generic Resources – Fundamentals of Product Description and Support".

4.1 Introduction

In different types of units there are four basic cases:

- Basic SI-units, which cover a number of fundamental units of mainly physical quantities defined by ISO-1000+A1,1992 & 1998.; e.g. meter or millimeter as unit for length measure or square meter as a unit for area measure. The unit may have a scaling prefix (milli, kilo, etc.).
- Conversion based units, which can be derived (by direct scaling) from SI-units; e.g. inch which can be defined using SI-units for length measure, i.e. an inch is 25.4 millimeters.
- Derived units, which can be defined as a derivation or combination of a number of basics units together with their dimensional exponent in that unit, e.g. kg / m2.
- Context dependent units, which cannot be directly defined as conversion based unit using SI-units.

With regard to the usage of measure defined types (*IfcLengthMeasure*, *IfcTimeMeasure*, etc.) as attribute data types in the IFC object model, there two basic cases:

- The data type of an entity attribute is a measure defined type as such. In this case it's the global unit assignment for the the corresponding unit for this measure type that defines the unit for all the usages of this defined measure type. E.g.,

```
ENTITY IfcScheduleTimeControl;
   …
   ScheduleDuration: IfcTimeMeasure;
   …
END_TYPE;
```

- The data type of an attribute is *IfcMeasureWithUnit*, which allows for definition of unit per instance of that entity type, independent of global unit assignment; E.g.

```
ENTITY IfcConstructionMaterialResource;
   …
   OrderQuantity : IfcMeasureWithUnit;
   …
END_TYPE;
```

Note: Here the relevant measure defined type (from the IfcMeasureWithUnit.ValueComponent : IfcValue select list) is not exactly defined by the schema, but it should be stated elsewhere in the specification documentation.

In general, it is recommended not to mix different units for same measure defined types, if it can be avoided. Below some examples of each of the above cases are giving.

Note*: In the example instantiations in the form of IFC data exchange files, only the measure and unit -relevant attributes are given the values; the other attributes are given no values (in the form of $-sign) independent of whether they should actually have values because of being non-optional attributes.*

4.2 Global unit assignment

The global unit assignment of the project defines the global units for measures and values when the units are not otherwise defined more specifically using entity type *IfcMeasureWithUnit* as attribute's data type. The global unit assignment is a UoF, handled at the *IfcProject* (see 1.5.6).

The relevant entity within the *IfcMeasureResource*, which is used to list all globally defined units, is the *IfcUnitAssignment*.

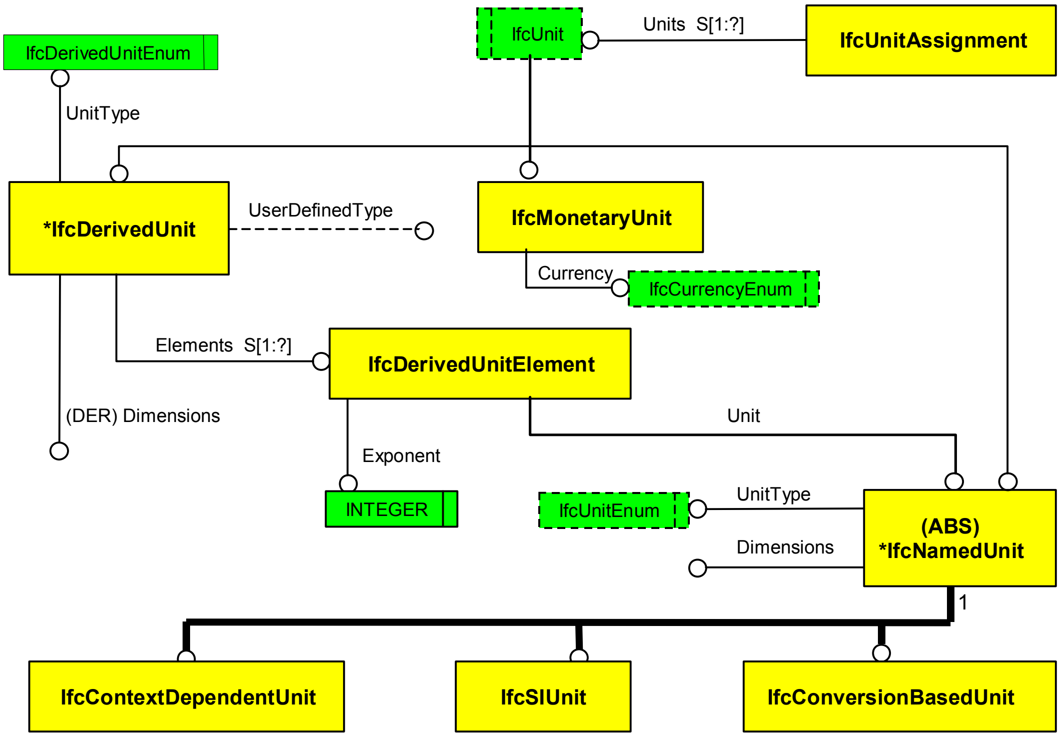

Figure 67 : Definition of IfcUnitAssignment and the IfcUnit select

4.2.1 Basic SI-units as global units

SI units are defined by the SI unit name, provided as an enumeration of all SI units, as defined in ISO 1000. A Prefix (such as milli, kilo, tera, etc.) can be provided in addition. If it is omitted, the basic SI unit is given.

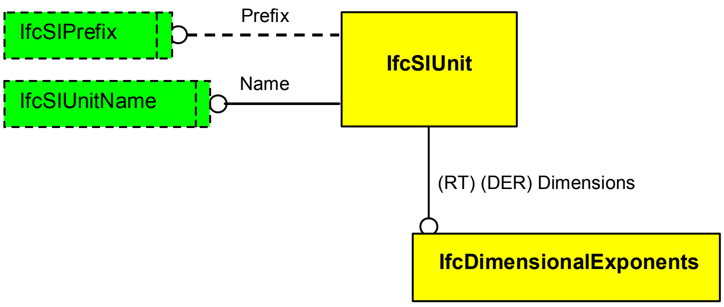

Figure 68 : Definition of IfcSIUnit

An example where project's global basic length, area and volume units are defined as SI units. Here all length measures are given as millimeter, all area measures as square meter and all volume measures as cubic meter.

```
#1= IFCPROJECT('fabcdeghijklmnopqrst02',#7,'test project',$,$,$,$,(#20),#30);
/* global units */
#30=IFCUNITASSIGNMENT((#33, #34, #35));
#33=IFCSIUNIT(*, .LENGTHUNIT., .MILLI., .METRE.);
#34=IFCSIUNIT(*, .AREAUNIT., $, .SQUARE_METRE.);
#35=IFCSIUNIT(*, .VOLUMEUNIT., $, .CUBIC_METRE.);
```

All measures within the geometric representations of IFC objects (see chapter 2) are given by measures with defined data types referring back to the global units, i.e. all lengths are given by the same unit (e.g. millimeter as in the example above).

Another example is the exchange of the global unit for angle measures (e.g. the plane angle measures for rotations). Here the unit is given by radian.

```
#1= IFCPROJECT('fabcdeghijklmnopqrst02',#7,'test project',$,$,$,$,(#20),#30);
#30=IFCUNITASSIGNMENT((#33, #34, #35, #36));
#33=IFCSIUNIT(*, .LENGTHUNIT., .MILLI., .METRE.);
#34=IFCSIUNIT(*, .AREAUNIT., $, .SQUARE_METRE.);
#35=IFCSIUNIT(*, .VOLUMEUNIT., $, .CUBIC_METRE.);
#36=IFCSIUNIT(*. .PLANEANGLEUNIT., $, .RADIAN.);
```

4.2.2 Conversion based units as global units

Global units within a project could also be given as derived units (although the use of standard SI units is encouraged). In this case the *IfcUnitAssignment* should refer to the derived unit definition, given by the unit type, a name and the conversion rate in regard to the SI units for the same unit type.

If degrees should be used for plane angle measures, in contrary to the example above (i.e. 180' instead of 3.1416, or π), than it has to be declared as a derived unit, referring to the radian as the underlying SI unit.

```
#1= IFCPROJECT('fabcdeghijklmnopqrst02',#7,'test project',$,$,$,$,(#20),#30);
#30= IFCUNITASSIGNMENT((#33, #34, #35, #36));
#33= IFCSIUNIT(*, .LENGTHUNIT., .MILLI., .METRE.);
#34= IFCSIUNIT(*, .AREAUNIT., $, .SQUARE_METRE.);
#35= IFCSIUNIT(*, .VOLUMEUNIT., $, .CUBIC_METRE.);
#36= IFCCONVERSIONBASEDUNIT(#40, .PLANEANGLEUNIT., 'DEGREE', #41);
#40= IFCDIMENSIONALEXPONENTS(0, 0, 0, 0, 0, 0, 0);
#41= IFCMEASUREWITHUNIT(IFCPLANEANGLEMEASURE(57.29577951308232), #50);
#50= IFCSIUNIT(*. .PLANEANGLEUNIT., $, .RADIAN.);
```

4.2.2.1 Conversion based Imperial units as global units

An example where projects global basic length and area units are defined as imperial units (inches and square feet), which are further defined as conversion based units relative to SI units millimeter and square meter. In the example SI unit cubic meter is defined as the global volume unit:

```
#1=IFCPROJECT($, $, '', $, $, $, $, $, #2, $, $);
#2=IFCUNITASSIGNMENT((#6, #9, #5));
#3=IFCSIUNIT(*, .LENGTHUNIT., .MILLI., .METRE.);
#4=IFCSIUNIT(*, .AREAUNIT., $, .SQUARE_METRE.);
#5=IFCSIUNIT(*, .VOLUMEUNIT., $, .CUBIC_METRE.);
#6=IFCCONVERSIONBASEDUNIT(#8, .LENGTHUNIT., 'INCH', #7);
#7=IFCMEASUREWITHUNIT(IFCLENGTHMEASURE(25.40005), #3);
#8=IFCDIMENSIONALEXPONENTS(1, 0, 0, 0, 0, 0, 0);
#9=IFCCONVERSIONBASEDUNIT(#10, .AREAUNIT., 'SQUARE_FEET', #11);
#10=IFCDIMENSIONALEXPONENTS(2, 0, 0, 0, 0, 0, 0);
#11=IFCMEASUREWITHUNIT(IFCAREAMEASURE(0.0929), #4);
```

4.2.3 Derived units as global units

An example definition of a unit for specific heat capacity (Joule / kg Kelvin), which is defined as a derived unit based on basic SI units:

```
#1=IFCPROJECT($, $, $, $, $, $, $, $, #2, $, $);
#2=IFCUNITASSIGNMENT((#3));
#3=IFCDERIVEDUNIT((#5, #6, #4), .SPECIFICHEATCAPACITYUNIT., $);
#4=IFCDERIVEDUNITELEMENT(#7, 1);
#5=IFCDERIVEDUNITELEMENT(#8, -1);
#6=IFCDERIVEDUNITELEMENT(#9, -1);
#7=IFCSIUNIT(*, .ENERGYUNIT., $, .JOULE.);
#8=IFCSIUNIT(*, .MASSUNIT., .KILO., .GRAM.);
#9=IFCSIUNIT(*, .THERMODYNAMICTEMPERATUREUNIT., $, .KELVIN.);
```

4.2.4 Use of defined measure type with global unit assignment

An example where the attribute *IfcScheduletimeControl.ScheduleDuration* : *IfcTimeMeasure*, is of data type *IfcTimeMeasure* without possibility of using specific unit from *IfcMeasureWithUnit*. Then the global unit for time measure applies; in this case seconds (the example's 172800 seconds would be equivalent to 2 days):

```
#1=IFCPROJECT($, $, $, $, $, $, $, $, #2, $, $);
#2=IFCUNITASSIGNMENT((#3));
#3=IFCSIUNIT(*, .TIMEUNIT., $, .SECOND.);
#4=IFCSCHEDULETIMECONTROL($, $, $, $, $, $, $, $, $, $, $, $, $, $, 172800., $, $, $, $,
   $, $, $);
```

4.3 Local unit assignment

An example where the *IfcConstructionMaterialResource.OrderQuantity* : *IfcMeasureWithUnit* of data type *IfcMeasureWithUnit* is given a value in liters (value 10 liters e.g. as a quantity value for paint) as a conversion based unit based on SI unit cubic meter. Here an "overriding" specific unit can be defined on the instance level independent of the global unit for volume measure (here SI unit cubic meter):

```
#1=IFCPROJECT($, $, '', $, $, $, $, $, #2, $, $);
#2=IFCUNITASSIGNMENT((#8));
#3=IFCCONSTRUCTIONMATERIALRESOURCE($, $, $, $, $, $, $, $, $, $, $, $, #4, $, $);
#4=IFCMEASUREWITHUNIT(10., #5);
#5=IFCCONVERSIONBASEDUNIT(#6, .VOLUMEUNIT., 'LITRE', #7);
#6=IFCDIMENSIONALEXPONENTS(3, 0, 0, 0, 0, 0, 0);
#7=IFCMEASUREWITHUNIT(IFCVOLUMEMEASURE(0.001), #8);
#8=IFCSIUNIT(*, .VOLUMEUNIT., $, .CUBIC_METRE.);
```

5 Material Definitions

5.1 Associating material definition with Objects in IFC model

Objects in IFC model that can have an associated material definition. All instances of subtypes of the *IfcElement* (with the exception of *IfcOpeningElement*) can have an instantiated inverse relationship '*HasAssociations'* pointing to *IfcRelAssociatesMaterial*. This provides a pointer to *IfcMaterialSelect*, where the actual instance is either

- of type *IfcMaterial* (to be used in case of one single solid material),
- of type *IfcMaterialList* (for multiple material elements when precise structure is not specified), or
- of type *IfcMaterialLayerSetUsage* (for layered elements, where the structure is specified).

In the IFC model the material association is defined as (fully attributed view):

```
ENTITY IfcRelAssociatesMaterial;
   (* ENTITY IfcRoot *)
      GlobalId :  IfcGloballyUniqueId;
      OwnerHistory :  IfcOwnerHistory;
      Name : OPTIONAL IfcLabel;
      Description : OPTIONAL IfcText;
   (* ENTITY IfcRelAssociates *)
      RelatedObjects  : SET [1:?] OF IfcRoot;
   (* ENTITY IfcRelAssociatesMaterial *)
      RelatingMaterial : IfcMaterialSelect;
END_ENTITY;

TYPE IfcMaterialSelect = (
      IfcMaterial,
      IfcMaterialList,
      IfcMaterialLayerSetUsage);
END_TYPE;
```

The material association relationship allows to assign the same material definition to multiple instances of *IfcElement*, or *IfcTypeObject*, and thereby it allows to share the same material information across many objects. Therefore the material information (such as material layer sets, material lists and materials) can be stored and exchanges separately from the actual building elements.

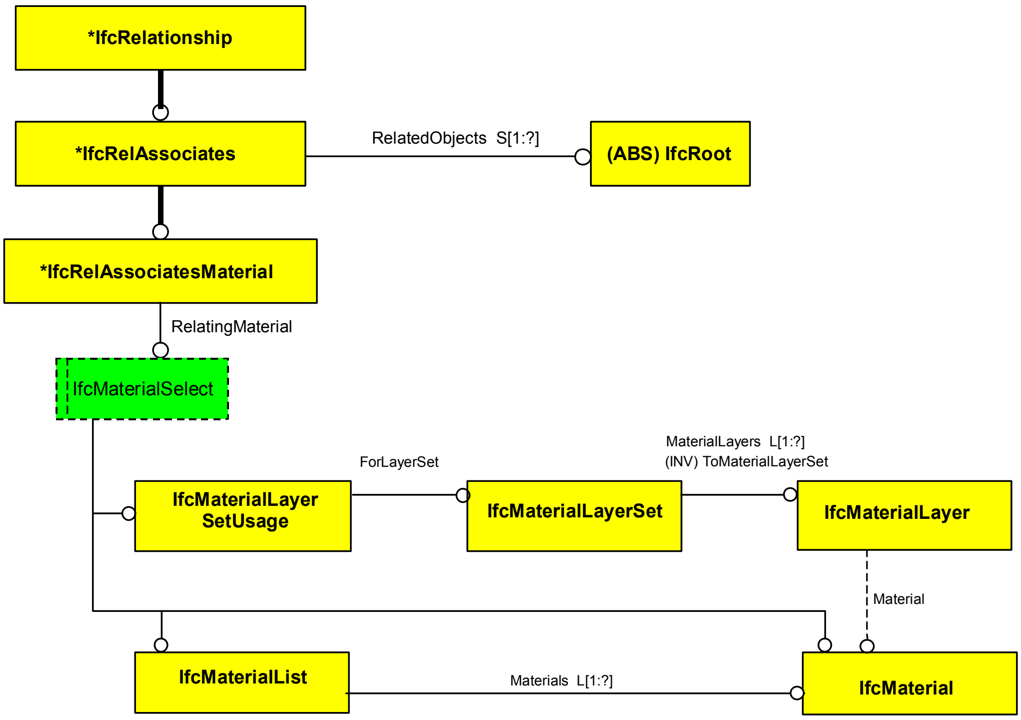

Figure 69 : Definition of material assignement

5.1.1 Associating a single material

In the first case, where there is a solid element, e.g. a concrete wall, the select item *IfcMaterial* would be used. This assigns the same material to the complete body of the element. The result is an isotropic material definition for the element.

```
ENTITY IfcMaterial;
      Name : IfcLabel;
   INVERSE
      ClassifiedAs : SET [0:1] OF IfcMaterialClassificationRelationship FOR
                     ClassifiedMaterial;
END_ENTITY;
```

Example:

Three concrete column are exchanged by referencing the same material definition.

```
#11=IFCCOLUMN('abcdefghijklmnopqrst11',#21,$,$,$,$,$,$);
#12=IFCCOLUMN('abcdefghijklmnopqrst12',#22,$,$,$,$,$,$);
#13=IFCCOLUMN('abcdefghijklmnopqrst13',#23,$,$,$,$,$,$);
#3=IFCRELASSOCIATESMATERIAL('abcdefghijklmnopqrst02',#24,$,$,(#11,#12,#13),#4);
#4=IFCMATERIAL('Concrete');
```

5.1.2 Associating a list of materials

In the second case, where there is an element made of several materials, e.g. a wall of concrete, brickwork and mineral wall insulation, but the exact structural configuration is not given, the select item *IfcMaterialList* would be used:

Note: The use of material list for walls is only allowed for arbitrary walls (i.e. instances of IfcWall, which are not instances of IfcWallStandardCase).

```
ENTITY IfcMaterial;
      Materials : LIST OF [1:?] IfcMaterial;
END_ENTITY;
```

Example:

The IFC file could appear for the case of the arbitrary wall as:

```
#1=IFCWALL('abcdefghijklmnopqrst01',#21,$,$,$,$,$,$);
#3=IFCRELASSOCIATESMATERIAL('abcdefghijklmnopqrst02',#22,$,$,(#1),#4);
#4=IFCMATERIALLIST(#5,#6,#7);
#5=IFCMATERIAL('Concrete');
#6=IFCMATERIAL('Brick');
#7=IFCMATERIAL('Mineral wool');
```

5.1.3 Associating material layers

In the third case, where there is an element made of several materials, e.g. a layered wall made of concrete, brickwork and mineral wall insulation, and the exact structural configuration is given, the select item *IfcMaterialLayerSetUsage* should be used. This also allows to reflect the material layers within the presentation of the element:

```
ENTITY IfcMaterialLayerSetUsage;
      ForLayerSet  : IfcMaterialLayerSet;
      LayerSetDirection  :  IfcLayerSetDirectionEnum;
      DirectionSense  :  IfcDirectionSenseEnum;
      OffsetFromReferenceLine  : IfcLengthMeasure;
END_ENTITY;

TYPE IfcLayerSetDirectionEnum = ENUMERATION OF (AXIS1, AXIS2, AXIS3);
END_TYPE;

TYPE IfcDirectionSenseEnum = ENUMERATION OF (POSITIVE, NEGATIVE);
END_TYPE;
```

Here the relationship '*ForLayerSet*' points to the instance of *IfcMaterialLayerSet* that is used to describe the material information of this particular wall. The attributes '*LayerSetDirection*', '*DirectionSense*' and '*OffsetFromReferenceLine*' give the orientation and location of the layer set relative to the wall geometry:

Note, that one instance of IfcMaterialLayerSet can be used by several walls, i.e. the same material combination may be assigned with different offsets to several walls.

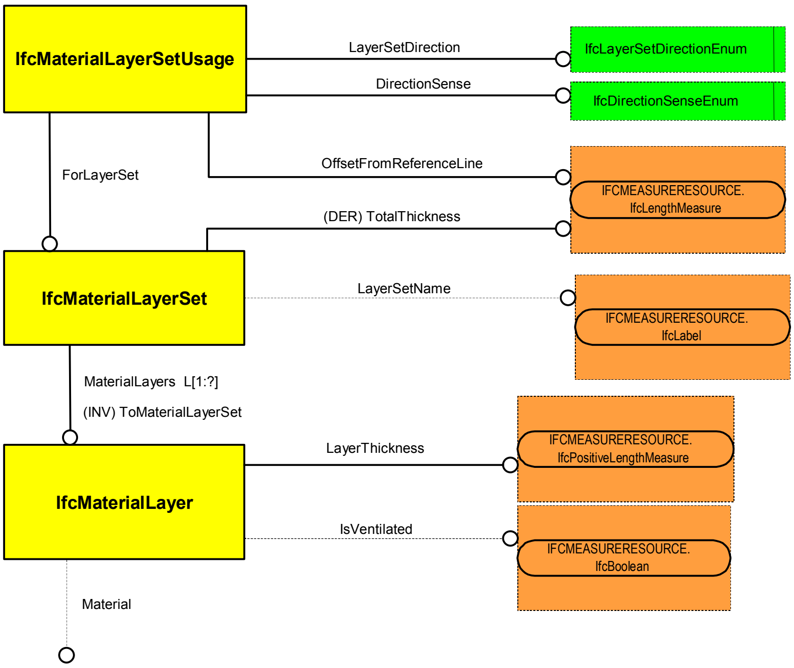

Figure 70 : Definition of material layer set

For a layered wall, the *IfcMaterialLayerSet.TotalThickness* shall be equal to the wall thickness (measured in the Y dimension of the extrusion coordinate system), and the layer set Y direction shall coincide with the extrusion profile positive or negative Y-direction (depending on the '*DirectionSense*' attribute). Thus, the attribute '*LayerSetDirection*' shall be set to AXIS2. The reference line shall be the wall axis path.

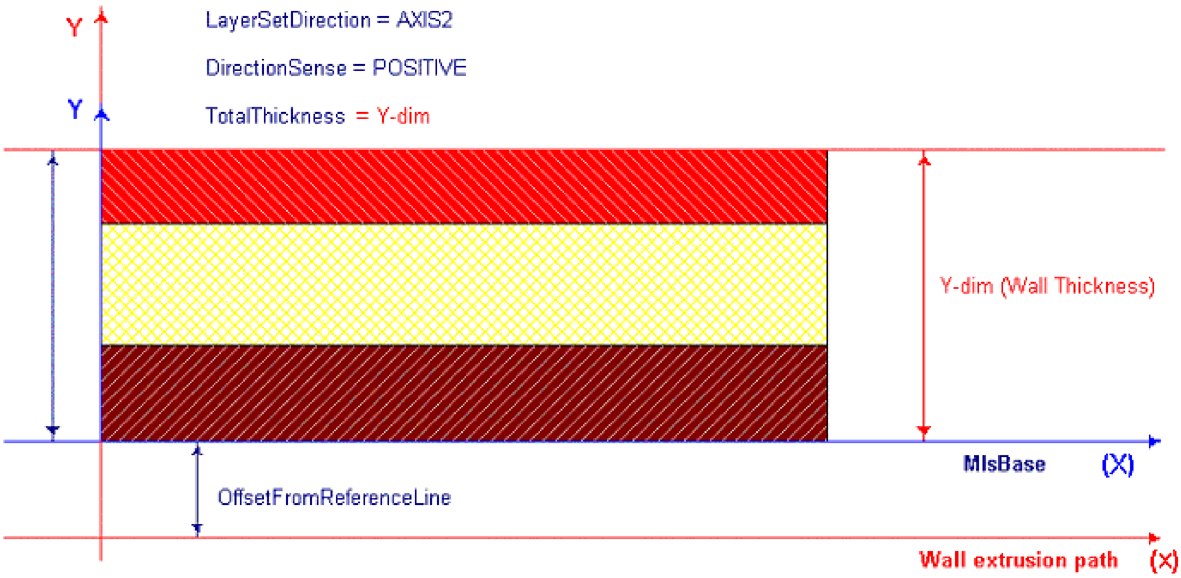

Figure 71 : Usage of material layer set

Figure 71 shows the definition of the material layer set being outside of the wall path, starting from the '*OffsetFromReferenceLine'* (positive offset) and assigning all layers into the same positive direction.

Example

In a case, where a wall (and its material layer set) with vertically extruded geometry has been assigned centric to the wall path, the IFC file could be as:

```
#1=IFCWALL('abcdefghijklmnopqrst01',#21,$,$,$,$,$,$);
#3=IFCRELASSOCIATESMATERIAL('abcdefghijklmnopqrst02',#22,$,$,(#1),#4);
#4=IFCMATERIALLAYERSETUSAGE(#5,.AXIS2.,.POSITIVE.,-150.);
#5=IFCMATERIALLAYERSET((#6,#7,#8),'Layered wall type 1');
#6=IFCMATERIALLAYER(#9,100.,$);
#7=IFCMATERIALLAYER(#10,120.,$);
#8=IFCMATERIALLAYER(#11,80.,$);
#9=IFCMATERIAL('Concrete');
#10=IFCMATERIAL('Mineral wool');
#11=IFCMATERIAL('Brick');
```

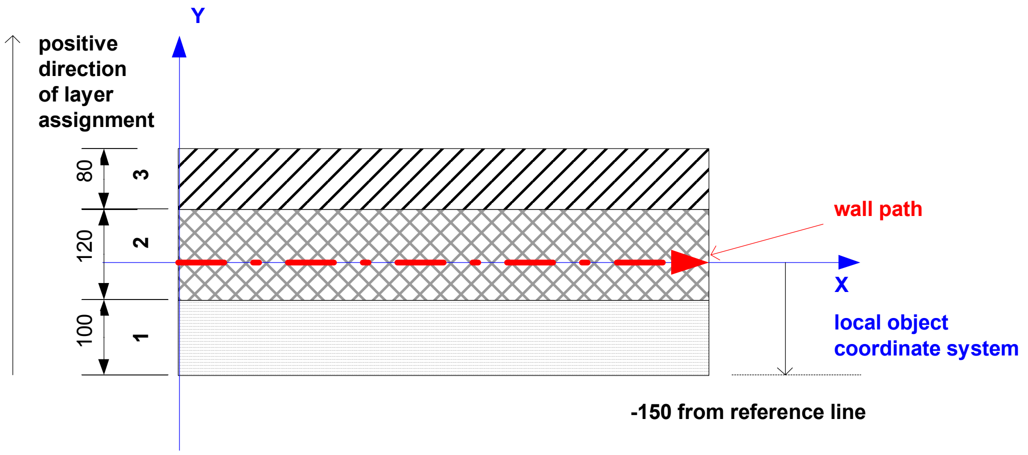

Figure 72 : Example for material layer set use assigned to a wall

5.2 Material Classifications

Classifications of a material can be expressed using general *IfcClassificationReference* (see the discussion about the concept of classification in 6.2), associated with specific material through *IfcMaterialClassificationRelationship*, defined as:

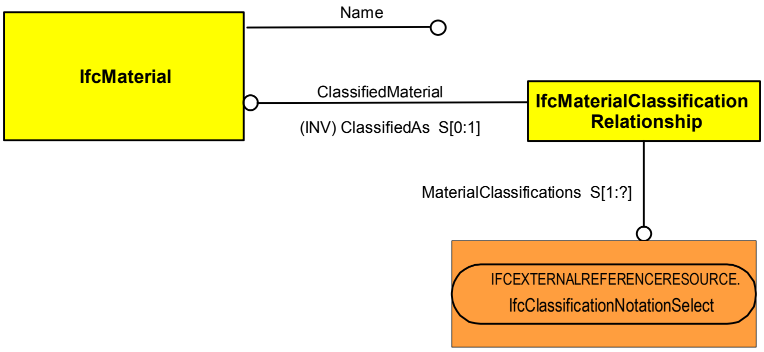

Figure 73 : Definition of Material Classification

Example:

In case of a particular steel material "S275JR" (formerly "Fe430B") according to Euronorm EN10025 (1993), this could appear in the exchange file as:

```
#20=IFCMATERIAL('Steel');
#30=IFCMATERIALCLASSIFICATIONRELATIONSHIP((#40),#20);
#40=IFCCLASSIFICATIONREFERENCE ($,'S275JR','Fe430B',#41);
#41=IFCCLASSIFICATION('CEN','1',$,'EN10025');
```

6 Referencing External Information

The IFC Model allows to reference information, which is externally defined. In this case only the access information is stored within the IFC exchange set (mostly the URL for HTTP access protocol), and eventually some header information (e.g. the name of the reference, a short summary, etc.). The actual content however remains at the source location.

There are three types of external references within the IFC2x model:

- document references
- classification references
- library references

6.1 Referencing External Documents

to be added after version 1.0 of the document

6.2 Referencing External Classifications

It is recognized that there are many different classification systems in use throughout the AEC/FM industry and that their use differs according to geographical location, industry discipline and other factors. For a generic model such as IFC, it is necessary to allow for the adoption of any rational classification system whether it is based on elements, work sections or any other classifiable division.

The classification model is able to represent classifications according to the most advanced current concepts from work in ISO TC59, ICIS (International Construction Information Society) and EPIC (European Product Information Coding) as well as more traditional classifications such as those in the various SfB forms used internationally, CAWS, Master format etc., or other national classification system, like the DIN in Germany.

The classification model forms part of the *IfcExternalReferenceResource* schema and specifies the use of the independent resources necessary for the scope and information requirements for the exchange and sharing of classification information between application systems. Such information may be used at all stages of the life cycle of a building.

The following are within the scope of the classification model in IFC2x:

- The provision of one or more classification notations to an object.
- The inclusion of one or more facets to a classification notation.
- Referencing of facets of a classification notation from a described source (classification item or classification table)
- Exposure of the hierarchy of a classification structure.
- Identification of the source of the classification.
- The designation of a classification in terms of its source, edition and name.
- The provision of a means of semantically identifying the meaning of a classification notation.
- Referencing a classification held on an external source.

The following is out of scope of the classification model in IFC2x:

- The ability to translate from one classification notation to another.

6.2.1 Concept of Associating Classification to Objects

Objects are classified by attaching a classification (either light weight classification reference or a fully defined facet within a classification table) by means of a relationship instance. This relationship class, *IfcRelAssociatesClassification*, forms part of the *IfcKernel* schema. It is used to apply an *IfcClassificationReference or IfcClassificationNotation* to an object or an object type and property set through its relation to *IfcRoot* (from which all objects that can be classified are subtyped).

Each instance of *IfcRelAssociatesClassification* enables the association of one classification notation with one or many objects. See also section 1.6.2.1 and 6.2.7 for the concept of association of classifications.

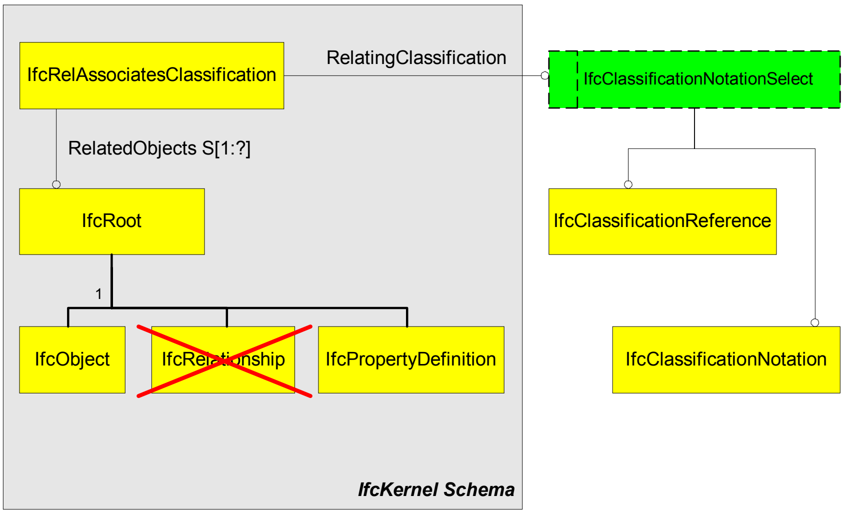

Figure 74 : Definition of associating a classifcation

NOTE: By virtue of a where rule at the IfcRelAssociates, relationships are prohibited from having external references or definitions.

It is possible for an object to have multiple classifications. This is achieved through the use of multiple instances of *IfcRelAssociatesClassification*, each instance pointing to a particular classification notation and to the objects that are classified by this notation.

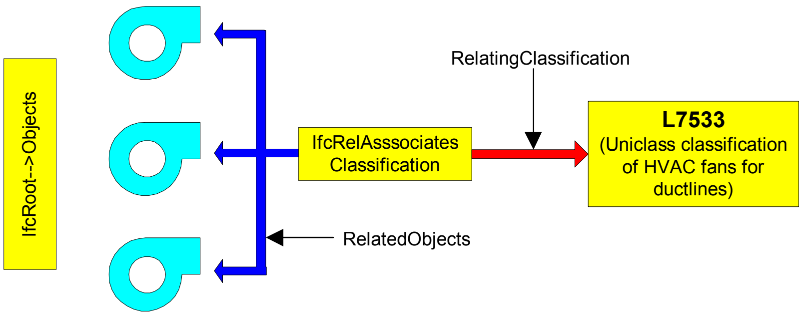

Figure 75 : Example of attaching a classification to multiple instances

Using the *IfcRelAssociatesClassification* relationship class means that any object that is not classified does not have to carry empty attributes for classification (as was the case with previous versions of the IFC model). This also facilitates a better subdivision of the IFC2x model into implementation views.

6.2.2 Concept of Classification Notation

The principal applied is that any type of object can be classified. No distinction is made between a product, a process, a control or a resource. The means by which an object may be classified is the *IfcClassificationNotation* class. An *IfcClassificationNotation* is a notation used from published reference (which may be either publicly available from a classification society or is published locally for the purposes of an organization, project or other purpose).

Note that a classification notation may be developed using various notation facets. A facet is a part of the actual notation but which has a specific meaning. For instance, it may be appropriate to classify an item by owning actor (represented by A=Architect) and by an entry from a classification table such as CI/SfB (represented by 210 for external wall). This gives a classification as A210.

All classifications of an object that are contained within the IFC model are made through the *IfcClassificationNotation* class. For a given object, the *IfcRelAssociatesClassification* class makes the connection between the *IfcObject* (abstract superclass, here: *IfcDoor* has been taken as an example for instantiation) and the *IfcClassificationNotation* (used for full classification).

IfcObject <-- IfcRelAssociatesClassification --> IfcClassificationNotation.

Example:

Consider an object that is to be classified with the notation "L6814". In this case, the IfcRelAssociatesClassification has the form:

```
#100=IFCDOOR ('gabcdeghijklmnopqrst02',#1000,.....);
#200=IFCCLASSIFICATIONNOTATIONFACET ('L6814');
#300=IFCCLASSIFICATIONNOTATION ((#200));
#400=IFCRELASSOCIATESCLASSIFICATION ('gabcdeghijklmnopqrst02',#1001,$,$,(#300),#100);
```

If the object is to have other notations (e.g. B2186 and Z6793), i.e. there are multiple classifications available to an object, then the IfcRelAssociatesClassification has the form:

```
#100=IFCDOOR ('gabcdeghijklmnopqrst02',#1000,.....);
#200=IFCCLASSIFICATIONNOTATIONFACET ('L6814');
#210=IFCCLASSIFICATIONNOTATIONFACET ('B2186');
#220=IFCCLASSIFICATIONNOTATIONFACET ('Z6793');
#300=IFCCLASSIFICATIONNOTATION ((#200,#210,#220));
#400=IFCRELASSOCIATESCLASSIFICATION('gabcdeghijklmnopqrst02',#1001,$,$,(#300),#100);
```

It is a requirement that a classification notation can only bring together facets from the same classification system or source. Bringing together notation facets from different sources within the same classification notation is not allowed. However, since the *IfcRelAssociatesClassification* class does allow for multiple classifications to a single object, then it is possible to define multiple classification notations where each notation contains facets from a single source.

Example:

Consider an object that is to be classified with the notations from two distinct classification systems. In this case, the IfcRelAssociatesClassification has the form:

```
#100=IFCDOOR ('gabcdeghijklmnopqrst02',#1000,.....);
#200=IFCCLASSIFICATIONNOTATIONFACET ('L6814');
#210=IFCCLASSIFICATIONNOTATIONFACET ('B2186');
#220=IFCCLASSIFICATIONNOTATIONFACET ('Z6793');
#300=IFCCLASSIFICATIONNOTATIONFACET ('106');
#400=IFCCLASSIFICATIONNOTATIONFACET ('A-31-623');
#410=IFCCLASSIFICATIONNOTATIONFACET ('B-62-562');
#500=IFCCLASSIFICATIONNOTATION ((#200,#210,#220));
#510=IFCCLASSIFICATIONNOTATION ((#300));
#520=IFCCLASSIFICATIONNOTATION ((#400,#410));
#600=IFCRELASSOCIATESCLASSIFICATION ('gabcdeghijklmnopqrst02',#1001,$,$,
      (#500,#510,#520),#100);
```

6.2.3 Concept of Classification Notation Facet

Many modern classification systems (such as Uniclass, BSAB etc.) allow for multi-faceted classification. This is reflected in the classification model through provision of the *IfcClassificationNotationFacet* class where each instance represents one facet of a classification notation. The classification notation itself is then made up of a list of notation facets. The list is declared to be a unique list to ensure both that there is order in the specification of the notation facets and to ensure that a facet can only be included once in each list.

An *IfcClassificationNotationFacet* object holds an individual classification value that is to be assigned to an object through *IfcClassificationNotation* and *IfcRelAssociatesClassification* objects. An *IfcClassificationNotationFacet* is a group of alphanumeric characters used within a classification notation.

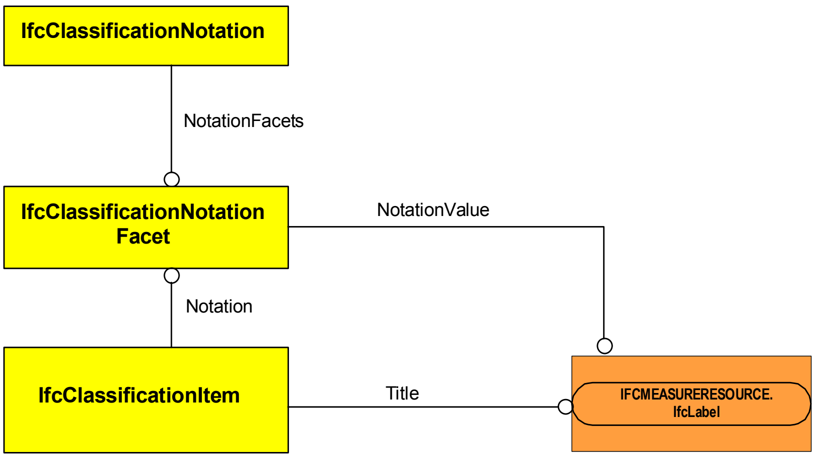

Figure 76 : Definition of a classification facets and items

Example:

> *For instance, considering a fan in an HVAC ductline. As a construction product, it has the Uniclass notation facet L7533. However, if we consider a work section view with the fan as part of the low velocity air conditioning system, the notation facet is JU30. Therefore, there are two facets in the classification notation which becomes ('JU30', 'L7533')*

6.2.4 Concept of Classification Item

An *IfcClassificationItem* is a class of classification notations used. Note that the term 'classification item' is used in preference to the more usual (but deprecated) term 'table' for improved flexibility. For example, the classification item "L681" in Uniclass may be used to contain all subsequent notation facets within that class of classifications which has the title "Proofings, insulation"(e.g. L6811, L6812, L6813 etc.).

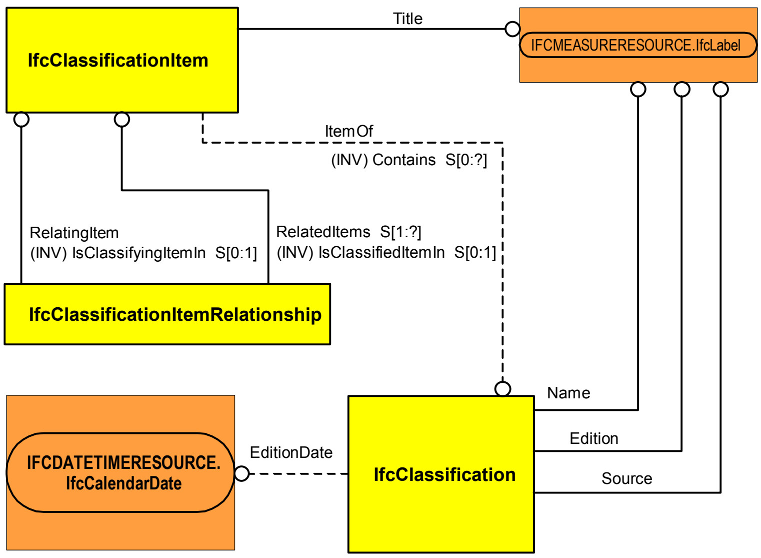

Figure 77 : Definition of classification system

6.2.5 Concept of Classification Item Relationship

Classification systems are generally declared in a hierarchical structure in which a facet at an upper level in the hierarchy (parent level) is a generalization of the facets at the next lower level in the hierarchy (child level). The hierarchy of the classification system is defined in tables (ref. CI/SfB) or sections (ref. CAWS) or some other named, coherent approach. Typically, the position in the hierarchy is identified by a nomenclature or label e.g. X12, and the identity at the child level is derived by adding (concatenating) characters e.g. X121, X122, X123 etc.

Within the classification model, the ability to identify location in a classification hierarchy is termed the *IfcClassificationItem* (so as not to be identified as table, section etc. but as a generalizations of these terms).

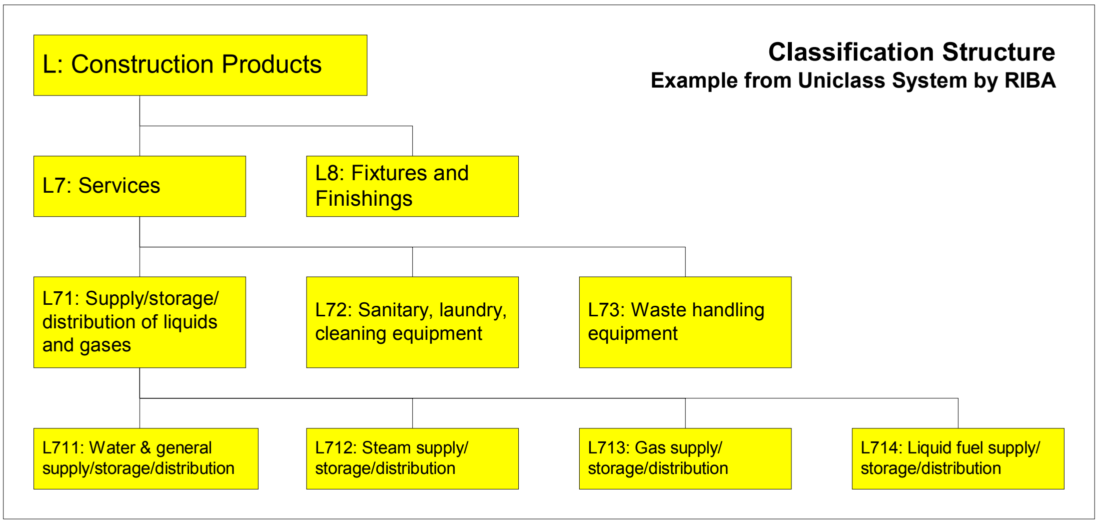

Figure 78 : Example of a hierarchical classification system

The whole of the classification hierarchy can now be exposed through the IFC Model. This is achieved by providing the recursive relationship class *IfcClassificationItemRelationship*. As a further restriction; a classification hierarchy cannot contain any instance of *IfcClassificationItem* more than once.

An *IfcClassificationItemRelationship* is a relationship class that enables the hierarchical structure of a classification system to be exposed through its ability to contain related classification items and to be contained by a relating classification item. *IfcClassificationItem's* can be progressively decomposed using the *IfcClassificationItemRelationship* such that the relationship always captures the information about the parent level (relating) item and the child level (related) items of which there can be many. The following example shows how this could be achieved for the Uniclass system.

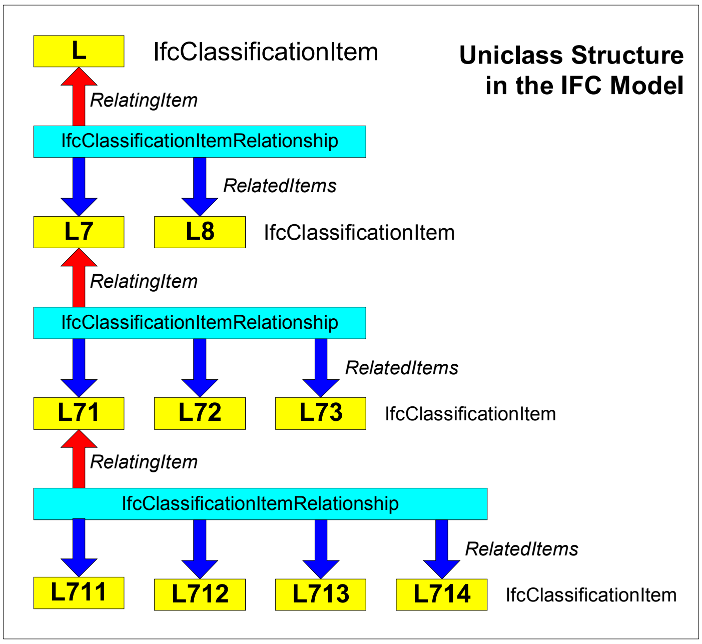

Figure 79 : Example of building a classification system in IFC2x

The inverse relationships from *IfcClassificationItem* to *IfcClassificationItemRelationship* enable information about the relationship to be recovered by the items concerned so that they are also aware of the decomposition. The cardinality of the inverse relationship is that an *IfcClassificationItem* can be the classifying item in maximum one relationship and can be a classified item in maximum one relationship. This reflects typical classification approaches that use strict hierarchical decomposition (or taxonomy) and do not have matrix relationships.

Example:

The example used in Figure 79 shows a decomposition detail from the UNICLASS system. This can be seen from:

```
#200=IFCCLASSIFICATIONITEM (#100,#2,'Door Panel');
#201=IFCCLASSIFICATIONITEM (#101,#2,'Door Panel');
#202=IFCCLASSIFICATIONITEM (#102,#2,'Door Panel');
#203=IFCCLASSIFICATIONITEM (#103,#2,'Door Panel');
#204=IFCCLASSIFICATIONITEM (#104,#2,'Door');
#205=IFCCLASSIFICATIONITEMRELATIONSHIP (#204,(#200,#201,#202,#203));
```

6.2.6 Concept of Classification System

Each classification item belongs to an *IfcClassification*. This provides the means to identify the classification system being used. *IfcClassification* has attributes that define its source, edition and name.

- *Name* - identifies what would usually be considered to be the name of the classification system such as CI/SfB, BSAB, CAWS, Masterformat, Uniformat etc.
- *Edition* - provides a version identification.
- *EditionDate* - identifies the date at which the edition became operational.
- *Source* - identifies the publishing reference of the classification system (e.g. for Uniclass, the source would be RIBA)

An *IfcClassification* is used for the arrangement of objects into a class or category according to a common purpose or their possession of common characteristics.

Example:

The objective is to minimize the number of IfcClassification objects contained within a populated IFC model. Ideally, each classification system or source used should have only one IfcClassification object. However, because multiple classification is allowed, there may be many IfcClassification objects used, each identifying a different classification system or source. An example of the use of the IfcClassification class, defining a single classification item is:

```
#1=IFCCALENDARDATE (31,8,1997);
#2=IFCCLASSIFICATION('RIBA','1',#1,'Uniclass');
#3=IFCCLASSIFICATIONITEM (#104,#2,'Door');
```

6.2.7 Concept of Classification Referencing

Recognizing that it may not be appropriate to maintain the whole detail of classification within the object model, and taking the view that information may well be referenced from external sources by its address to enable access through mechanisms such as the World Wide Web, the classification model of IFC2x includes a means to reference a classification through the *IfcClassificationReference* class. This is a subtype of *IfcExternalReference* that has a label (which can be the reference address) and identifier. Additional information may be available concerning the classification system source that is being referenced.

An *IfcClassificationReference* is a reference into a classification system or source (see *IfcClassification*). An optional inherited "*ItemReferenced*" key is also provided to allow more specific references to classification items (or tables) by type.

6.2.7.1 Lightweight Classification

The *IfcClassificationReference* can be used as a form of 'lightweight' classification through the 'ItemReference' attribute inherited from the abstract *IfcExternalReference* class. In this case, the 'ItemReference' could take (for instance) the Uniclass notation "L6814" which, if the classification was well understood by all parties and was known to be taken from a particular classification source, would be sufficient. This would remove the need for the overhead of the more complete classification structure of the model.

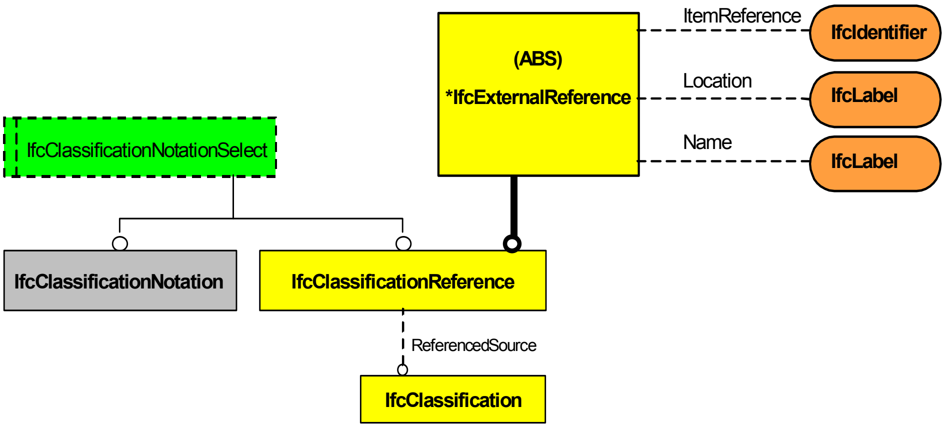

Figure 80 : Definition of a classification reference

However, it is not recommended that this lightweight method be used in cases where more than one classification system is in use or where there may be uncertainty as to the origin or meaning of the classification.

6.2.7.2 Referencing from an External Source

Classifications of an object may be referenced from an external source rather than being contained within the IFC model. This is done through the *IfcClassificationReference* class.

Example:

Consider the Uniclass notation "L6814" which has the title "Tanking". In this case, the optional attribute 'ItemReference' uses the notation "L6814", and the attribute "Name" uses the title 'Tanking' that would otherwise be applied to the IfcClassificationItem (if it was to be contained in the model).

The location of the classification reference may be found from a classification server that is available via the Internet, like http://www.ncl.ac.uk/classification/uniclass/tanking.htm#6814.

```
#1=IFCCALENDARDATE(31,8,1997);
#2=IFCCLASSIFICATION('RIBA','1',#1,'Uniclass');

#100=IFCDOOR('gabcdeghijklmnopqrst01',#1001, ..... );
#300=IFCRELASSOCIATESCLASSIFICATION('gabcdeghijklmnopqrst02',#1001,$,$,(#200),#100);

#200=IFCCLASSIFICATIONREFERENCE
('http://www.ncl.ac.uk/classification/uniclass/tanking.htm#6814','L6814','Tanking',#2);
```

Because the relation between *IfcRelAssociatesClassification* and classification is actually made at the *IfcClassificationNotationSelect* class that allows classification to be either contained (full internal classification) or referenced (lightweight classification), it is possible to assign both contained and referenced classifications to an object.

```
#1=IFCCALENDARDATE (31,8,1997);
#2=IFCCLASSIFICATION ('RIBA','1',#1,'Uniclass');

#100=IFCDOOR ('gabcdeghijklmnopqrst01',#1001,.....);
#500=IFCRELASSOCIATESCLASSIFICATION('gabcdeghijklmnopqrst02',#1001,$,$,(300,#400),#100);

#200=IFCCLASSIFICATIONNOTATIONFACET ('L6814');
#210=IFCCLASSIFICATIONNOTATIONFACET ('B2186');
#220=IFCCLASSIFICATIONNOTATIONFACET ('Z6793');
#300=IFCCLASSIFICATIONNOTATION ((#200,#210,#220));
#400=IFCCLASSIFICATIONREFERENCE
('http://www.ncl.ac.uk/classification/uniclass/tanking.htm#6814','L6814','Tanking',#2);
```

6.2.8 Translation Between Classification Notations

Whilst several different classification notations may be applied to an object, there is no equivalence between the classification notations USED. Therefore, the IFC Classification Model cannot be used to translate from one classification notation to another. The reason for this is that, generally, it is possible to select from any of several different notations within a classification system for an object. The actual selection is the responsibility of the user according to circumstances. Therefore, there is a many to many relationship between classification systems for which there is no resolution at this stage of development.

6.3 Referencing External Libraries

It is anticipated that many property sets defined externally to the IFC Model will be referenced from a library or database of product data held by a manufacturer/supplier, an information provider acting on their behalf or some other body holding significant information sources.

Referencing information from a library allows for the data to be retained in the library rather than in the IFC Model. In an information exchange/sharing scenario, this can be useful since it means that the amount of information needing to be transferred can be minimized. The IFC Model provides a structure[4] that enables property sets either to be referenced from the library in which they are held or delivered from the library into the IFC model as a property definition that can be associated with one or more objects.

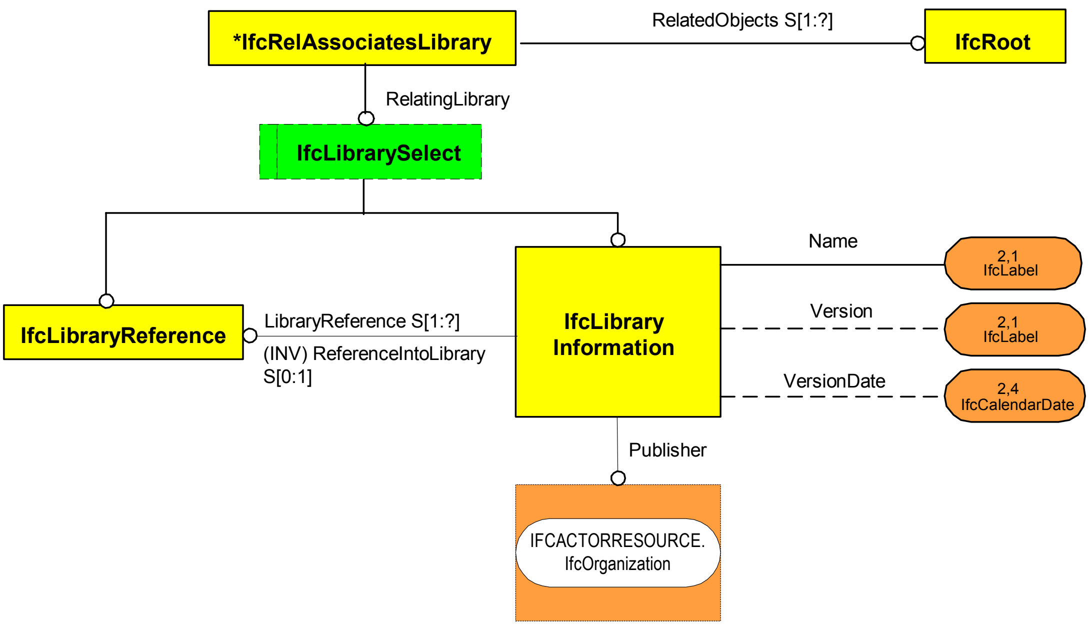

Figure 81 : The mechanism to reference libraries

The relationship class *IfcRelAssociatesLibrary* manages the linkage between the library and an object. It should be noted that the related objects in this case are instances of the IfcRoot class. This is because a selection of whether to associate the library with an *IfcObject* or an *IfcPropertyDefinition* has to be made. In this case, this cannot be done through the use of a SELECT data type since the EXPRESS language does not allow the declaration of inverse relationships from a SELECT type. Therefore, the association has to be made to the common supertype of *IfcObject* and *IfcPropertyDefinition* which is *IfcRoot*. Since *IfcRoot* also has other subtypes, a rule is applied to the relationship class which constrains the subtypes of *IfcRoot* that can have a library associated.

In selecting the *IfcLibraryReference* to be used, the attributes of the *IfcExternalReferencResource* class are inherited. These enable the location of the library to be captured. Location can be any fully qualified address such as a directory path on a CD. However, it is anticipated that it will more usually be a URL enabling referencing to occur across the World Wide Web.

[4] Although this discussion relates to the referencing of property definitions, the model structure discussed can also manage the association of object information from libraries.

An *ItemReference* may also be captured that enables a pointer into the library source to be captured. This provides a more complete identity for the information within the library.